Introduzione

Problema decisionale

- Solitamente affrontato da un *decisore*, con atteggiamento *informale*
- Una alternativa che può portare ad un considerevole miglioramento delle prestazioni è l'uso di un atteggiamento *formale*, che richiede la formalizzazione del problema tramite un *modello*

La domanda da porsi per valutare l'opportunità di adottare un attegiamento formale è: "Può avere senso provare ad usare un modello?"

Generalmente può avere senso quando:

- Si ritiene che le decisioni prese con atteggiamento informale possano essere migliorate sensibilmente
- Si ritiene di avere o di poter acquisire una sufficiente conoscenza dei costi, profitti e vincoli corrispondenti alle possibili decisioni

Questo spesso accade quando il problema da affrontare è ben definito ed il numero di possibili alternative è enorme (ad esempio: *turnazione del personale*)

Nel cercare di costruire un modello (non solo in ambito decisionale, ma in un qualunque ambito ingegneristico) è fondamentale tenere presente che:

- Il modello non *rappresenta* la realtà ma la *schematizza* (è una *semplificazione, astrazione* della realtà)
- Solo i modelli che si sanno *risolvere* sono di utilità pratica

È quindi essenziale definire un "buon" *compromesso* tra il *livello di astrazione* del modello, individuando gli aspetti essenziali del problema decisionale (generalmente per "approssimazioni successive"), e la *complessità* della sua risoluzione, che richiede una conoscenza dei modelli che si è *attualmente* in grado di risolvere

Si vuole sempre un modello *significativo e risolubile*, altrimenti l'atteggiamento formale non serve

A loro volta i modelli possono essere classificati in

- *Qualitativi* (o *logici*) che esprimono solo relazioni di causa-effetto senza utilizzare grandezze numeriche

- *Quantitativi* (o *matematici*) che, oltre ad esprimere relazioni logiche di causa-effetto, utilizzano equazioni e disequazioni per esprimere i legami tra i loro parametri – che a lora volta possono essere classificati in:
  - *variabili*, che possono essere cambiati (e che quindi corrispondono alle *decisioni* possibili)
  - *costanti*, che non possono essere cambiati, ulteriormente classificati in:
    * *deterministici*, dei quali si suppone noto il valore
    * *stocastici* (o *aleatori*) che possono assumere più valori secondo una data *distribuzione di probabilità*

È fondamentale notare come la distinzione tra variabili e costanti così come tra deterministici e stocastici debba essere di tipo *operativo*, da decidere nella definizione del modello tenendo presente il compromesso tra significatività e risolubilità

Quali modelli si usano nella pratica?

Attualmente (e si ha l'impressione che la situazione rimarrà invariata almeno per alcune decine di anni), gli approcci formali a problemi di interesse pratico tramite modelli quantitativi utilizzano prevalentemente i seguenti due tipi di modelli:

- Modelli di *simulazione* per capire il funzionamento di un sistema, validando o invalidando eventuali decisioni prese (in altre parole, la simulazione non dice quali decisioni prendere ma indica se un insieme di possibili decisioni va bene o meno)
- Modelli di *Programazione Lineare Intera (ILP)* per avere l'indicazione di quali decisioni prendere, generalmente considerando una situazione (alquanto) più semplificata rispetto ai modelli di simulazione

I risultati ottenuti tramite la soluzione di un modello ILP sono generalmente validati tramite la simulazione, che indica anche che il modello deve essere rivisto qualora il comportamento della soluzione "ottima" del modello ILP risulti essere "cattivo" per la simulazione

L'indiscusso successo dei modelli ILP è legato prevalentemente a due fattori concomitanti:

- L'efficacia dei metodi risolutivi di tipo euristico e branch-and-bound basati sul rilassamento continuo di *Programmazione Lineare (LP)*, dovuta prevalentemente all'efficacia dei metodi risolutivi per modelli LP, prevalentemente basati sull'Algoritmo del Simplesso
- La flessibilità di tali modelli, che consentono di esprimere vincoli di varia natura (logici, di capacità, ...) e nonlinearità di varia natura (saturazione di costi, prodotto/ massimo/ minimo di variabili, ...)

Struttura del Corso

Considerando che la descrizione dei modelli di simulazione nel corso di *Fondamenti di Ricerca Operativa LA* è ampiamente sufficiente, il corso approfondisce la parte di modelli ILP, con particolare enfasi su modelli con un numero "esponenziale" di vincoli e variabili (come sono molto frequentemente quelli di interesse pratico) e sulla loro soluzione, presentando applicazioni di turnazione del personale, instradamento di veicoli, taglio/impaccamento bi- e tri-dimensionale e sequenziamento di lavorazioni

L'esame consta di una parte scritta obbligatoria e, facoltativamente, di una parte orale – il voto dello scritto è in 30-simi e l'orale lo può modificare al più di 3 punti in basso o in alto

Il voto finale non ha limite di validità, e la sua registrazione è possibile solo durante le registrazioni che seguono ciascun appello, la cui data viene comunicata assieme ai risultati dell'appello stesso

I testi sono gli appunti delle lezioni e queste dispense, mentre *per eventuale consultazione* si può fare riferimento al testo:

G. Ghiani, R. Musmanno, *Modelli e Metodi per l'Organizzazione dei Sistemi Logistici*, Pitagora, Bologna

Si sconsiglia in ogni caso di acquistare tale testo ai soli fini del superamento dell'esame

Richiami e Complementi di Ricerca Operativa

Programmazione Lineare e Dualità

Un generico problema di *Programmazione Lineare (LP)* ha la forma:

$$\min \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m_1 \tag{2}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = m_1 + 1, \ldots, m \tag{3}$$

$$x_j \geq 0, \quad j = 1, \ldots, n_1 \tag{4}$$

dove gli scalari n, n_1, m, m_1, i vettori $c = (c_j) \in \mathbb{R}^n$ e $b = (b_i) \in \mathbb{R}^m$ e la matrice $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ sono *costanti note* mentre il vettore $x = (x_j) \in \mathbb{R}^n$ corrisponde alle *variabili decisionali* delle quali deve essere determinato il valore ottimo

(Si noti che le variabili x_j per $j = n_1 + 1, \ldots, n$ possono anche assumere valori negativi)

Un qualsiasi problema di ottimizzazione in cui la funzione obiettivo sia lineare nelle variabili ed i vincoli siano equazioni o disequazioni lineari nelle variabili può essere espresso nella forma sopra, eventualmente:

- cambiando segno alla funzione obiettivo se di massimo
- cambiando verso alle disuguaglianze che sono nella forma "$\leq$"

La presenza delle disuguaglianze rende il problema non risolubile con metodi standard di algebra lineare (che sarebbero applicabili se ci fossero solo equazioni e le variabili non fossero vincolate ad essere ≥ 0), richiedendo algoritmi appropriati quali l'Algoritmo del Simplesso

Nella teoria della Programmazione Lineare e, ai fini di questo corso, nella risoluzione di modelli con un numero esponenziale (o comunque molto elevato) di variabili, è essenziale il concetto di *Problema Duale* di un LP

Il *duale* del generico LP indicato sopra (chiamato *primale*) ha la forma:

$$\max \sum_{i=1}^{m} b_i y_i \tag{5}$$

$$\sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j = 1, \ldots, n_1 \tag{6}$$

$$\sum_{i=1}^{m} a_{ij} y_i = c_j, \quad j = n_1 + 1, \ldots, n \tag{7}$$

$$y_i \geq 0, \quad i = 1, \ldots, m_1 \tag{8}$$

con vettore delle variabili $y = (y_i) \in \mathbb{R}^m$

In altre parole, il problema duale, di massimo, è ottenuto dal problema primale, di minimo:

- "scambiando" variabili e vincoli
- scambiando costi e termini noti
- trasponendo la matrice A dei vincoli
- facendo corrispondere ad ogni disuguaglianza una variabile ≥ 0
- facendo corrispondere ad ogni equazione una variabile che può anche assumere valori negativi

(ricordandosi che la regola di definizione sopra richiede che le disuguaglianze del primale siano nella forma "$\geq$" e definisce il duale con disuguaglianze nella forma "$\leq$")

È facile verificare che il duale del duale è un problema identico al primale di partenza, a meno del cambiamento di segno di equazioni e/o di variabili che possono assumere valori negativi

La proprietà fondamentale del problema duale è il seguente *teorema della dualità forte*:

Teorema 1 *Il valore della soluzione ottima del primale coincide con quella del duale, assumendo che entrambi siano ammissibili e limitati*

In particolare, il valore di ogni soluzione del duale è minore o uguale del valore di ogni soluzione ammissibile del primale

Questo implica che, date una soluzione primale x^* ed una soluzione duale y^*, esse sono *entrambe ottime se e solo se* $\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$

In altre parole, è possibile *certificare* l'ottimalità di una soluzione di un LP mostrando una soluzione del problema duale dello stesso valore, senza dover "rendere conto" di come la soluzione primale sia stata ottenuta

Questa "certificazione" dell'ottimalità indipendente dal procedimento di soluzione, unitamente all'esistenza di un problema duale che soddisfa il Teorema 1, *non è possibile per ILP*

Programmazione Lineare Intera e Mista

Un problema ILP è la *variante* di un problema LP in cui si è imposto il vincolo ulteriore che tutte le variabili siano intere:

$$x_j \qquad \text{intero}, \quad j = 1, \ldots, n \tag{9}$$

Un problema di *ILP Mista* è la *generalizzazione* di LP ed ILP in cui solo un sottoinsieme delle variabili devono essere intere:

$$x_j \qquad \text{intero}, \quad j \in S \tag{10}$$

per un qualche sottoinsieme $S \subseteq \{1, \ldots, n\}$

Entrambi i problemi sono risolubili, e sono nella pratica risolti, tramite algoritmi *branch-and-bound* in cui viene risolto il rilassamento continuo, ottenuto rimuovendo i vincoli di interezza

Grafi

I grafi sono utilizzati per rappresentare graficamente un grande numero di problemi finiti, aiutandone considerevolmente la comprensione

A seconda della natura del problema da rappresentare possono essere usati grafi *non orientati* ed *orientati*

Grafi non orientati

Un grafo non orientato $G = (V, E)$ è definito da una coppia di insiemi: l'insieme V dei *vertici* (o *nodi*) e l'insieme E dei *lati*, ciascuno dei quali corrisponde ad una coppia di vertici

Data una coppia di vertici $i, j \in V$, il lato e corrispondente è indicato con $e = (i, j) \in E$, ed i, j sono detti gli *estremi* di e

Dato un vertice $i \in V$, l'insieme dei lati contenenti i (o con un estremo in i) è indicato con $\delta(i)$

Più in generale, dato un sottoinsieme di vertici $S \subseteq V$, l'insieme dei lati con esattamente un estremo in S è indicato con $\delta(S)$ (tracciando una linea chiusa che racchiuda i soli vertici in S, questi sono i lati che intersecano la linea)

Dato un sottoinsieme di vertici $S \subseteq V$, l'insieme dei lati con entrambi gli estremi in S è indicato con $E(S)$

Un grafo non orientato si dice *completo* se il suo insieme di lati contiene tutte le $|V|(|V|-1)/2$ possibili coppie di vertici

Grafi orientati

Un grafo orientato $G = (V, A)$ è definito da una coppia di insiemi: l'insieme V dei *vertici* (o *nodi*) e l'insieme A degli *archi*, ciascuno dei quali corrisponde ad una coppia *ordinata* di vertici

Data una coppia ordinata di vertici $i, j \in V$, l'arco a corrispondente è indicato con $a = (i, j) \in A$, i è detto la *coda* di a e j la *testa* di a

Dato un vertice $i \in V$, l'insieme degli archi con coda i è indicato con $\delta^+(i)$, e l'insieme degli archi con testa i è indicato con $\delta^-(i)$

Più in generale, dato un sottoinsieme di vertici $S \subseteq V$, l'insieme degli archi con coda in S e testa in $V \setminus S$ è indicato con $\delta^+(S)$, e l'insieme degli archi con testa in S e coda in $V \setminus S$ è indicato con $\delta^-(S)$

Dato un sottoinsieme di vertici $S \subseteq V$, l'insieme degli archi con entrambi gli estremi (testa e coda) in S è indicato con $A(S)$

Un grafo orientato si dice *completo* se il suo insieme di lati contiene tutte le $|V|(|V|-1)$ possibili coppie ordinate di vertici

Cenni di Teoria della Complessità

Complessità degli Algoritmi

La *complessità* di un *algoritmo* è una misura del tempo necessario per la sua esecuzione

Per prescindere dallo specifico esecutore dell'algoritmo, tale misura non si esprime come tempo fisico, ma come *numero di istruzioni elementari*, corrispondenti a confronti, assegnazioni, somme, sottrazioni, moltiplicazioni e divisioni

Inoltre, tale numero di istruzioni viene espresso come funzione della *dimensione dell'input* dell'algoritmo stesso, limitandosi al *caso peggiore* e fornendo solo l'*ordine di grandezza* della funzione, tramite la simbologia $O(\cdot)$:

Date due funzioni $f, g : \mathbb{N} \to \mathbb{N}$, si ha $f(n) = O(g(n))$ se esiste una costante k tale che $f(n) \leq kg(n)$ per ogni $n \in \mathbb{N}$ (in altre parole $g(n)$ è un limite superiore o *upper bound* di

$f(n)$ a meno di una costante moltiplicativa)

Talvolta, può essere utile la simbologia $\Omega(\cdot)$: date due funzioni $f, g : \mathbb{N} \to \mathbb{N}$, si ha $f(n) = \Omega(g(n))$ se esiste una costante k tale che $f(n) \geq kg(n)$ per ogni $n \in \mathbb{N}$ (in altre parole $g(n)$ è un limite inferiore o *lower bound* di $f(n)$ a meno di una costante moltiplicativa)

La dimensione dell'input, qui indicata con $|I|$, corrisponde al *numero di bit* necessari a codificare l'input stesso – dato che questo è direttamente proporzionale, ad esempio, al numero di caratteri ASCII necessari a codificare l'input stesso, e le costanti moltiplicative non sono considerate nella notazione $O(\cdot)$, si può immaginare la dimensione dell'input come questo numero di caratteri, con i numeri rappresentati in base 10

Si osservi che il numero di cifre necessarie a codificare un numero $n \in \mathbb{N}$ in base 10 è pari a $\lfloor \log_{10} n \rfloor + 1$

Esempio 1 Si consideri il problema di ordinare un vettore $a = (a_j)$ di n numeri in senso decrescente

L'algoritmo più semplice per tale problema determina il massimo elemento del vettore, lo inserisce in prima posizione, successivamente determina il secondo massimo elemento e lo inserisce in seconda posizione, e così via

Vengono eseguite n iterazioni, nella j-esima delle quali viene cercato il massimo tra $n-j+1$ elementi, eseguendo $n - j + 1$ confronti ed *al più* $n - j + 1$ assegnazioni

Tralasciando per semplicità le istruzioni di inizializzazione, nel caso peggiore il numero di istruzioni è $\sum_{j=1}^{n} 2(n - j + 1) = n(n + 1) = O(n^2)$

(Si noti che il tempo di esecuzione è anche, ad esempio, $O(n^3)$, oppure $O(2^n)$, in quanto la notazione $O(\cdot)$ fornisce un *upper bound* a meno di costanti moltiplicative, ma ovviamente la stima $O(n^2)$ è più precisa)

Per quanto riguarda la dimenzione dell'input $|I|$, se questo è fornito come sequenza "$n, a_1, \ldots, a_n$", con i valori separati da "," ed i valori codificati in base 10, questo è pari a

$$n + (\lfloor \log_{10} n \rfloor + 1) + \sum_{j=1}^{n} (\lfloor \log_{10} a_j \rfloor + 1) = \Omega(n)$$

Quindi, esprimendo il tempo di esecuzione come funzione di $|I|$, si può dire che questo è $O(|I|^2)$, o *quadratico* nella dimensione dell'input

Gli algoritmi sono classificati in due categorie principali, a seconda della loro complessità:

- Algoritmi *polinomiali*, la cui complessità è $O(|I|^m)$ per una qualche costante m
- Algoritmi *esponenziali*, la cui complessità è $\Omega(p^{|I|})$ per una qualche costante $p \in \mathbb{R}_+$

Per molti algoritmi esponenziali la complessità è dell'ordine di $2^{|I|}$, mentre per molti altri è sensibilmente peggiore, ad esempio $|I|!$ o $|I|^{|I|}$

Si noti che la complessità di alcuni algoritmi polinomiali potrebbe non essere indicata come funzione polinomiale, ma in ogni caso essere limitata superiormente da una funzione polinomiale – ad esempio gli algoritmi polinomiali più efficienti per il problema dell'ordinamento dell'Esempio 1 hanno complessità $O(n \log n)$

Inoltre, si osservi che in generale esistono algoritmi la cui complessità non è nè polinomiale nè esponenziale, ad esempio dell'ordine di $|I|^{\log_2 |I|}$, ma data la loro rarità è naturale aspettarsi che, dato un nuovo algoritmo da classificare, esso appartenga ad una delle due categorie sopra

Complessità dei Problemi

I problemi si classificano in due categorie in base alla loro complessità:

- Problema *polinomiale*, se esiste un algoritmo polinomiale che lo risolve
- Problema *NP-completo* (o *NP-difficile*, o *NP-hard*) se non si conosce alcun algoritmo polinomiale per risolverlo e se, qualora un tale algoritmo esistesse, allora esisterebbe per tutti i problemi NP-completi (per cui si *sospetta fortemente* che non esista)

Per quanto concerne la classificazione sopra, è fondamentale notare che:

- Quella data non è ovviamente la definizione di NP-completo (non fornita in questo corso) ma un'implicazione della definizione
- Non tutti i problemi appartengono ad una delle due categorie, ma le eccezioni sono estremamente rare, per cui è *naturale*, quando ci si trova di fronte ad un nuovo problema, chiedersi se sia polinomiale o NP-completo
- Non è difficile inventarsi degli algoritmi esponenziali per problemi polinomiali (ad esempio per l'ordinamento: genera tutte le $n!$ permutazioni del vettore a e verifica se sono ordinate in senso decrescente), ma questo *non deve far concludere* che il problema non sia polinomiale
- "Si sospetta fortemente che non esista", per gli amici "P vs NP", è uno dei problemi fondamentali della matematica attuale, vecchio quasi come il docente del corso e indicato tra i "Millennium Problems" `http://www.claymath.org/millennium/`

L'approccio ingegneristico quando ci si trova di fronte ad un problema da risolvere è il seguente:

- Chiedersi se il problema sia polinomiale o NP-completo

- Se è polinomiale, utilizzare un algoritmo polinomiale per risolverlo, dato che nella pratica gli esponenti del polinomio sono tipicamente *piccoli*, generalmente compresi tra 1 e 3
- Se è NP-completo, considerare che va maneggiato con cura, senza escludere a priori la possibilità di risolverlo perché gli algoritmi noti, di tipo esponenziale, funzionano malissimo nel *caso peggiore* ma potrebbero andare bene nel *caso medio* riferito alle istanze da risolvere

La stragrande maggioranza dei problemi di interesse pratico che non sono banalmente polinomiali risultano essere NP-completi

In questo corso ci si occupa di modelli ILP per molti problemi NP-completi di interesse pratico

LP, ILP e Complessità

Data la rilevanza pratica di LP ed ILP, è fondamentale conoscerne la complessità

Per quanto riguarda LP, il seguente risultato è stato dimostrato poco dopo l'inizio della teoria della complessità (primi anni '70)

Teorema 2 *L'algoritmo del simplesso è esponenziale*

Questo è un classico esempio dei limiti della teoria della complessità, che si limita a considerare il caso peggiore, in quanto in pratica (o se si preferisce "nel caso medio") l'algorimo e' veloce

In ogni caso, la complessità di LP è rimasta aperta fino al 1979, quando è stato dimostrato

Teorema 3 *LP è un problema polinomiale*

Il risultato ha messo in luce un'altro limite della teoria della complessità, in quanto l'algoritmo polinomiale che implica il teorema (*algoritmo dell'ellissoide*) non è mai stato competitivo con quello del simplesso

In ogni caso, il fatto che LP fosse polinomiale ha stimolato la ricerca di altri algoritmi di soluzione polinomiali, alcuni dei quali (che vanno sotto il nome generico di *metodi a punto interno*) sono competitivi con l'algoritmo del simplesso

Allo stato attuale, il vantaggio clamoroso dell'algoritmo del simplesso su metodi alternativi è il fatto di essere in grado, in modo del tutto semplice e naturale, di ripartire dalla soluzione precedente (e non da capo) una volta che alcuni vincoli o variabili siano stati aggiunti al problema

Per quanto riguarda ILP, praticamente tutti i problemi combinatori NP-completi sono formulabili come ILP, il che implica

Teorema 4 *ILP è un problema NP-completo*

Il fatto che ILP sia più difficile di LP è quindi messo in luce dalla teoria della complessità, e giustifica il fatto che, per risolvere ILP, si utilizzi un metodo che risolve degli LP come sottoproblemi

LP vs ILP: Formulazioni Ideali

Ricordandosi che l'algoritmo del simplesso determina sempre un vertice ottimo, è facile dimostrare il seguente fatto

Osservazione 1 *Ogni ILP è risolubile come LP applicando l'algoritmo del simplesso*

Dim. Considerando le soluzioni di un ILP, corrispondenti a punti interi che soddisfano un insieme di disequazioni ed equazioni lineari, si consideri la *chiusura convessa* di tali soluzioni, definita dall'insieme di tutte le loro combinazioni convesse (ovverosia combinazioni lineari con coefficienti non-negativi che sommano ad 1)

La chiusura convessa è a sua volta un poliedro in cui tutti i vertici corrispondono a soluzioni dell'ILP, quindi applicando l'algoritmo del simplesso a tale chiusura convessa si determina la soluzione ottima dell'ILP □

Esempio 2 Si consideri il problema ILP la cui regione ammissibile è definita da

$$\{3x_1 + 2x_2 \leq 6,\ 2x_1 + 3x_2 \leq 6,\ x_1, x_2 \geq 0,\ \text{intero}\}$$

L'insieme delle soluzioni dell'ILP è dato da $\{(0,0),\ (0,1),\ (0,2),\ (1,0),\ (1,1),\ (2,0)\}$, e la corrispondente chiusura convessa dal poliedro

$$\{x_1 + x_2 \leq 2\ x_1, x_2 \geq 0\}$$

Mentre risolvendo il rilassamento continuo della formulazione iniziale la soluzione ottima è frazionaria, ad esempio, con funzione obiettivo $\max x_1 + x_2$, per qualsiasi funzione obiettivo le soluzioni determinate dall'algoritmo del simplesso sulla chiusura convessa sono intere

La formulazione corrispondente alla chiusura convessa è detta talvolta *formulazione ideale* di un ILP

Per ILP corrispondenti a problemi polinomiali, *praticamente in tutti i casi* è nota una formulazione ideale, il che rende LP una sorta di "re" dei problemi polinomiali in quanto praticamente tutti sono riconducibili ad esso

Per ILP corrispondenti a problemi NP-completi, invece, si incontrano i seguenti ostacoli nella costruzione di una formulazione ideale:

- Il numero di disuguaglianze necessarie a descrivere la chiusura convessa è esponenziale nella dimensione dell'ILP
- Queste disuguaglianze sono "sconosciute", nel senso che è un problema NP-completo determinare se una data disuguaglianza è valida o meno per la chiusura convessa

Gli ostacoli sopra impediscono in pratica di determinare le formulazioni ideali per problemi NP-completi

D'altra parte la nozione di formulazione ideale suggerisce di cercare di formulare un dato ILP tramite disuguaglianze "forti" (quale $x_1 + x_2 \leq 2$ nell'esempio) e non tramite disuguaglianze "deboli" (quali $3x_1 + 2x_2 \leq 6$ e $2x_1 + 3x_2 \leq 6$ nell'esempio)

L'uso di disuguaglianze "forti" consente di ottenere un rilassamento continuo "vicino" alla chiusura convessa, garantendo buoni lower bound (per problemi di minimo) ed una veloce convergenza del branch-and-bound

Per la maggior parte dei problemi di interesse pratico, si possono derivare formulazioni ILP "deboli" con pessimi lower bound e tempi di branch-and-bound infiniti (da un punto di vista pratico) così come formulazioni ILP "forti" con ottimi lower bound e tempi di branch-and-bound accettabili

Il corso è ampiamente basato sulla derivazione di formulazioni ILP "forti" per problemi di interesse pratico, discutendo casi in cui è evidente quali siano disuguaglianze "deboli" e quali "forti", molto più che per l'esempio giocattolo sopra

Modelli ILP per Problemi NP-Completi di Base

Uncapacitated Facility Location

Nel problema dell'*Uncapacitated Facility Location* (UFLP) ("localizzazione" di centri di servizio), sono dati:

- m *clienti* da servire
- n *impianti* (o centri di servizio) che possono essere attivati o meno
- per ogni impianto j, f_j è il *costo di attivazione dell'impianto* j
- per ogni cliente i ed impianto j, c_{ij} è il *costo di servizio* del cliente i tramite l'impianto j

Il problema richiede di stabilire (i) quali impianti attivare e (ii) con quale degli impianti attivati servire ciascun cliente, in modo da minimizzare il costo totale di attivazione e servizio

Dato che, una volta stabiliti gli impianti attivati, converrà sempre servire ciascun cliente i con l'impianto ativato j per cui c_{ij} è minimo, la *vera* decisione è completamente rappresentata dalle variabili binarie:

$$y_j := \begin{cases} 1, & \text{se l'impianto } j \text{ è attivato} \\ 0, & \text{altrimenti} \end{cases}$$

D'altra parte, dato che tramite queste sole variabili non si riesce ad esprimere un modello ILP del problema, è necessario introdurre anche le variabili binarie:

$$x_{ij} := \begin{cases} 1, & \text{se il clilente } i \text{ è servito dall'impianto } j \\ 0, & \text{altrimenti} \end{cases}$$

Con queste variabili un modello "quasi" ILP è il seguente:

$$\min \sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{11}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m \tag{12}$$

$$x_{ij} = 1 \Rightarrow y_j = 1, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{13}$$

$$y_j, x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{14}$$

Ovviamente quello sopra non è un modello ILP in quanto i vincoli logici (13) devono essere espressi come equazioni o disuguaglianze lineari:

$$x_{ij} \leq y_j, \quad i = 1, \ldots, m, \ j = 1, \ldots, n \tag{15}$$

oppure

$$\sum_{i=1}^{m} x_{ij} \leq m y_j, \quad j = 1, \ldots, n \tag{16}$$

(Si noti che i secondi vincoli sono ottenuti sommando i primi per $i = 1, \ldots, m$)

Dato che entrambe (15) e (16) sono valide, si sarebbe tentati di preferire le seconde perché in numero notevolmente inferiore (n contro nm)

In realtà, il rilassamento continuo della formulazione con (16) è notevolmente più debole (o, se si preferisce, il rilassamento continuo della formulazione con (15) è notevolmente più vicino alla chiusura convessa delle soluzioni intere)

Esempio 3 Si consideri il caso banale in cui $n = m$, $f_j = 1$ e $c_{jj} = 0$ per $j = 1, \ldots, n$, $c_{ij} = +\infty$ per $i \neq j$

In questo caso la soluzione ottima di UFLP, coincidente con la soluzione ottima del rilassamento continuo della formulazione con (15), è data da $y_j = x_{jj} = 1$ per $j = 1, \ldots, n$ ed ha valore n

La soluzione ottima del rilassamento continuo della formulazione con (16) è invece data da $y_j = 1/n$ ed $x_{jj} = 1$ per $j = 1, \ldots, n$ ed ha valore 1

Si osservi infine come, per entrambe le formulazioni ILP, il vincolo che impone che le variabili x siano intere sia *ridondante*, nel senso che, per ogni y intero, conviene comunque porre $x_{ij} = 1$ per j tale che $c_{ij} = \min\{c_{ik} : y_k = 1\}$

Set Covering, Partitioning e Packing

Un caso particolare estremamente rilevante di UFLP, che in realtà corrisponde al problema di ottimizzazione combinatoria più importante nella pratica, si ottiene quando gli elementi della matrice c possono assumere i soli valori 0 e $+\infty$

(In altre parole, per ciascuna coppia cliente i e impianto j, o il cliente i può essere servito dall'impianto j a costo nullo, oppure il cliente i *non* può essere servito dall'impianto j)

In questo caso la formulazione si semplifica notevolmente in quanto il problema richiede "semplicemente" di determinare un sottoinsieme di impianti complessivamente con costi fissi minimi ed in grado di servire tutti i clienti

Definendo per ogni cliente i l'insieme degli impianti in grado di servirlo:

$$J_i := \{j : c_{ij} = 0\}, \quad i = 1, \ldots, m$$

il seguente è un modello ILP per il problema con le sole variabili y:

$$\min \sum_{j=1}^{n} f_j y_j \tag{17}$$

$$\sum_{j \in J_i} y_j \geq 1, \quad i = 1, \ldots, m \tag{18}$$

$$y_j \in \{0, 1\}, \quad j = 1, \ldots, n \tag{19}$$

Il problema si chiama *Set Covering*, e corrisponde ad un *generico* ILP in cui:

- tutte le variabili sono binarie
- tutti i vincoli sono disuguaglianze di tipo "$\geq$"
- tutti i termini noti sono pari a 1
- i coefficienti della matrice A dei vincoli sono binari

Più brevemente, utilizzando una notazione compatta matriciale Set Covering ha la forma:

$$\begin{aligned} \min\, & c^T x \\ Ax &\geq \mathbf{1} \\ x &\in \{0, 1\}^n \end{aligned} \tag{20}$$

dove $A \in \{0, 1\}^{m \times n}$, $\mathbf{1}$ è un vettore di m elementi uguali ad 1, e c^T indica il vettore c trasposto

Ci sono due varianti del set Covering anch'esse estremamente importanti in pratica

La prima, il *Set Partitioning*, si ottiene quando le disuguaglianze sono sostituite da equazioni:

$$\begin{aligned} \min\, & c^T x \\ Ax &= \mathbf{1} \\ x &\in \{0, 1\}^n \end{aligned} \tag{21}$$

La seconda, il *Set Packing*, si ottiene quando le disuguaglianze di tipo "$\geq$" sono sostituite da disuguaglianze di tipo "$\leq$" – in questo caso è naturale esprimere la funzione obiettivo nella forma di massimo:

$$\begin{array}{rcl} \max c^T x & & \\ Ax & \leq & \mathbf{1} \\ x & \in & \{0,1\}^n \end{array} \tag{22}$$

Si osservi che il Set Packing ha sempre la soluzione ammissibile $x = (0, \ldots, 0)$, che il Set Covering ha soluzione se e solo se $x = (1, \ldots, 1)$ è ammissibile, mentre non è ovvio verificare se esista una soluzione ammissibile di Set Partitioning – in realtà tale verifica risulta essere un problema NP-completo

A dispetto della piccola differenza formale, Set Covering e Set Packing sono problemi profondamente diversi:

- da un punto di vista pratico Set Covering è meno difficile e generalmente la sua formulazione è "forte" (oltre che non facilmente "rafforzabile")
- Set Packing ha un'interpretazione diretta come problema su grafo (discussa nel seguito e che Set Covering non ha) che indica chiarmente come "rafforzarne" la formulazione, che altrimenti risulta solitamente essere "debole"

Come osservazione finale, si noti come la condizione $x_j \in \{0, 1\}$ imposta sulle variabili dei tre problemi sopra sia equivalente a $x_j \geq 0,$ intero – in altre parole la condizione $x_j \leq 1$ nel rilassamento continuo dei tre problemi è *ridondante*

Capacitated Facility Location

Il problema del *Capacitated Facility Location* (CFLP) è la variante di UFLP in cui:

- ogni cliente i ha una *domanda* d_i
- ogni impianto j ha una *capacità* b_j, corrispondente alla quantità totale di domanda che può soddisfare

Il corrispondente modello ILP ottenuto considerando i vincoli (15) è il seguente:

$$\min \sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{23}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m \tag{24}$$

$$x_{ij} \leq y_j, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{25}$$

$$\sum_{i=1}^{m} d_i x_{ij} \leq b_j y_j, \quad j = 1, \ldots, n \tag{26}$$

$$y_j, x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{27}$$

Si osservi che, a differenza di UFLP, il vincolo che impone che le variabili x siano intere è necessario – in realtà esiste una versione di CFLP in cui la domanda di un cliente può essere soddisfatta da più impianti e quindi le variabili x possono essere frazionarie

I vincoli (25) non sono strettamente necessari nel modello ma possono "rafforzare" il rilassamento continuo del problema

Bin Packing e Knapsack

Un caso particolare di CFLP di interesse pratico si ha quando:

- i costi di servizio sono tutti nulli: $c_{ij} = 0,\ i = 1, \ldots, m,\ j = 1, \ldots, n$
- i costi di attivazione e le capacità degli impianti sono tutti uguali: $f_j = 1,\ b_j = b,\ j = 1, \ldots, n$

In questo caso, tutti gli impianti sono identici, e si deve attivare il *minimo numero* di impianti che consentano di soddisfare la domanda di tutti i clienti

Per completezza, si riporta il modello ILP corrispondente:

$$\min \sum_{j=1}^{n} y_j \tag{28}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m \tag{29}$$

$$x_{ij} \leq y_j, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{30}$$

$$\sum_{i=1}^{m} d_i x_{ij} \leq b y_j, \quad j = 1, \ldots, n \tag{31}$$

$$y_j, x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{32}$$

L'ambientazione tipica di questo caso particolare è differente: dati m *oggetti* con *pesi* $d_1, \ldots, d_m$ ed n *contenitori* (o *bin*) di *capacità* b, *impaccare* ciascun oggetto in un contenitore in modo che:

- per ciascun bin, il peso totale degli oggetti impaccati nel bin non superi la capacità
- il numero di bin utilizzati sia minimizzato

Questo è noto come il problema del *Bin Packing*

Assumeremo che $\sum_{i=1}^{m} d_i > b$ – in caso contrario il problema è banale

Un problema di base (più semplice) relativo ad un solo bin, in cui gli oggetti hanno anche dei *profitti* $p_1, \ldots, p_m$ e si vuole impaccare nel bin un sottoinsieme di oggetti di profitto massimo è noto come problema del *Knapsack*, ed ha il seguente modello ILP:

$$\max \sum_{i=1}^{m} p_i x_i \tag{33}$$

$$\sum_{i=1}^{m} d_i x_i \leq b \tag{34}$$

$$x_i \in \{0,1\}, \quad i = 1, \ldots, m \tag{35}$$

Nel corso di Fondamenti di Ricerca Operativa LA si è visto come sia semplice risolvere il rilassamento continuo del Knapsack

La stessa proprietà vale per il Bin Packing – definendo

$$\ell := \frac{\sum_{i=1}^{m} d_i}{b}$$

(che risulta essere un lower bound ovvio sul valore ottimo della soluzione) la seguente è una soluzione ottima del rilassamento continuo di (28)-(32) di valore ℓ:

$$y_j = \ell/n, \ j = 1, \ldots, n; \quad x_{ij} = 1/n, \ i = 1, \ldots, m, \ j = 1, \ldots, n$$

La discussione sopra mette in luce un primo chiaro svantaggio del modello ILP del Bin Packing visto: il rilassamento continuo del modello fornisce un lower bound ovvio, che in realtà, in alcuni casi, risulta anche essere "lontano" dal valore della soluzione ottima del bin packing

Un altro svantaggio è l'elevata *simmetria* del modello: per ogni soluzione intera di valore k, esistono $\binom{n}{k}k!$ soluzioni equivalenti – l'effetto pratico è che i vincoli di branching in un algoritmo branch-and-bound difficilmente aiutano ad ottenere una soluzione intera

Considerati gli svantaggi indicati sopra, è naturale chiedersi se il modello ILP visto per Bin Packing possa essere migliorato

In realtà, piuottsto che tentare di migliorare il modello dato, è meglio definire un nuovo modello (legato a quello visto, ma in modo non evidente e che non verrà discusso)

Il modello alternativo contiene un numero esponenziale (in m) di variabili, in quanto ogni variabile corrisponde ad un impaccamento ammissibile di oggetti in un bin

Formalmente, indicando con $\mathcal{S}'$ la collezione dei sottoinsiemi di oggetti che possono essere impaccati in un bin (il motivo per cui viene utilizzato l'apice sarà chiaro più avanti):

$$\mathcal{S}' := \left\{ S \subseteq \{1, \ldots, m\} : \sum_{i \in S} d_i \leq b \right\}$$

il nuovo modello contiene una variabile binaria per ogni $S \in \mathcal{S}'$:

$$x_S := \begin{cases} 1, & \text{se in soluzione c'è un bin contenente tutti e soli gli oggetti in S} \\ 0, & \text{altrimenti} \end{cases}$$

Il corrispondente ILP è di tipo Set Partitioning:

$$\begin{aligned} \min & \sum_{S \in \mathcal{S}'} x_S \\ & \sum_{S \in \mathcal{S}' : i \in S} x_S = 1, \quad i = 1, \ldots, m \\ & x_S \in \{0,1\}, \quad S \in \mathcal{S}' \end{aligned} \tag{36}$$

Si noti che il numero di variabili è $O(2^m)$, e quindi enorme per valori di m non troppo piccoli

Esempio 4 Si consideri il caso $m = 5$, $b = 10$, $d = (7, 5, 4, 4, 2)$, per il quale

$$\mathcal{S}' = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1,5\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,5\}, \{3,4,5\}\}$$

Il corrispondente modello di Set Partitioning ha la forma:

$$\begin{aligned} \min & \textstyle\sum_{S \in \mathcal{S}'} x_S \\ x_{\{1\}} + x_{\{1,5\}} &= 1 \\ x_{\{2\}} + x_{\{2,3\}} + x_{\{2,4\}} + x_{\{2,5\}} &= 1 \\ x_{\{3\}} + x_{\{2,3\}} + x_{\{3,4\}} + x_{\{3,5\}} + x_{\{3,4,5\}} &= 1 \\ x_{\{4\}} + x_{\{2,4\}} + x_{\{3,4\}} + x_{\{4,5\}} + x_{\{3,4,5\}} &= 1 \\ x_{\{5\}} + x_{\{1,5\}} + x_{\{2,5\}} + x_{\{3,5\}} + x_{\{4,5\}} + x_{\{3,4,5\}} &= 1 \\ x_S &\in \{0,1\}, \quad S \in \mathcal{S}' \end{aligned}$$

È possibile ridurre considerevolmente il numero di variabili del modello (che continua però ad essere esponenziale) nel seguente modo: si indichi con $\mathcal{S}$ la collezione dei sottoinsiemi *massimali* di oggetti che possono essere impaccati in un bin, cioè dei sottoinsiemi in $\mathcal{S}'$ ai quali non può essere aggiunto alcun altro oggetto senza violare il vincolo di capacità:

$$\mathcal{S} := \left\{ S \subseteq \{1, \ldots, m\} : \sum_{i \in S} d_i \leq b, \sum_{i \in S \cup \{j\}} d_i > b \text{ per ogni } j \notin S \right\}$$

Il nuovo modello contiene una variable x_S per ogni $S \in \mathcal{S}$ – è facile osservare che il modello (36) non è più valido sostituendo $\mathcal{S}'$ con $\mathcal{S}$ (come nel caso dell'esempio descritto)

D'altra parte, con le nuove variabili si ha un modello valido sostituendo le equazioni con disequazioni, ottenendo un modello di tipo Set Covering:

$$\begin{aligned} \min & \sum_{S \in \mathcal{S}} x_S \\ & \sum_{S \in \mathcal{S}: i \in S} x_S \geq 1, \quad i = 1, \ldots, m \\ & x_S \in \{0, 1\}, \quad S \in \mathcal{S} \end{aligned} \tag{37}$$

Osservazione 2 *Ad ogni soluzione del modello (36) corrisponde una soluzione del modello (37) dello stesso valore e viceversa*

Dim. Data una soluzione x^* del modello (36), una soluzione $\overline{x}$ del modello (37) con lo stesso valore si ottiene considerando ciascuna variabile $x^*_S = 1$, determinando un sottoinsieme massimale $\overline{S} \in \mathcal{S}$ tale che $S \subseteq \overline{S}$ e ponendo $\overline{x}_{\overline{S}} = 1$

Viceversa, data una soluzione $\overline{x}$ del modello (37), una soluzione x^* del modello (36) con lo stesso valore si ottiene inizializzando $I := \emptyset$, considerando, in un ordine arbitrario, ciascuna variabile $\overline{x}_S = 1$, e ponendo:

- $x^*_{S^*} = 1$ per $S^* := S \setminus I$
- $I := I \cup S^*$

□

In realtà si ha l'equivalenza completa anche per le soluzioni dei rilassamenti continui dei modelli (36) e (37), la cui dimostrazione è analoga

Il modello (37) è *migliore* del modello (36) perché:

- il suo numero di variabili è più piccolo (benchè esponenziale)
- i valori dei rilassamenti continui coincidono
- generalmente sono più facili da risolvere LP con disuguaglianze piuttosto che con equazioni

Esempio 5 Si consideri nuovamente il caso $m = 5$, $b = 10$, $d = (7, 5, 4, 4, 2)$

$$\mathcal{S} = \{\{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4, 5\}\}$$

Il corrispondente modello di Set Covering ha la forma:

$$\begin{array}{rcl} \min \sum_{S \in \mathcal{S}} x_S & & \\ x_{\{1,5\}} & \geq & 1 \\ x_{\{2,3\}} + x_{\{2,4\}} + x_{\{2,5\}} & \geq & 1 \\ x_{\{2,3\}} + x_{\{3,4,5\}} & \geq & 1 \\ x_{\{2,4\}} + x_{\{3,4,5\}} & \geq & 1 \\ x_{\{1,5\}} + x_{\{2,5\}} + x_{\{3,4,5\}} & \geq & 1 \\ x_S & \in & \{0, 1\}, \quad S \in \mathcal{S} \end{array}$$

Data la soluzione $x^*_{\{1,5\}} = x^*_{\{2,3\}} = x^*_{\{4\}} = 1$ del modello di (36), la corrispondente soluzione del modello (36) è $\overline{x}_{\{1,5\}} = \overline{x}_{\{2,3\}} = \overline{x}_{\{3,4,5\}} = 1$

Riassumendo, per il Bin Packing il modello ILP "naturale", certamente più semplice degli altri, ha molti svantaggi

Volendo quindi risolvere il problema tramite branch-and-bound, qualora il modello "naturale" non fornisca un risultato in tempi accettabili, occorre utilizzare il modello (37), da risolvere con i metodi illustrati più avanti nel corso

Il risultato che segue (senza dimostrazione) specifica il limite superiore al numero di sottoinsiemi in $\mathcal{S}$

Teorema 5 *Per un insieme di m oggetti il numero massimo di sottoinsiemi massimali di oggetti che possono essere impaccati in un bin è pari a $\binom{m}{m/2}$*

Utilizzando la formula di Stirling (alquanto precisa) per approsimare il fattoriale, si ha $\binom{m}{m/2} \approx 2^m/\sqrt{\pi m/2}$

Un esempio in cui il numero massimo è raggiunto si ottiene con m pari, $d_i = 1$ per $i = 1, \ldots, n$ e $b = m/2$, per cui $\mathcal{S}$ è formato da tutti i sottoinsiemi di $m/2$ oggetti

Stable Set e Clique

Si consideri un grafo non orientato $G = (V, E)$ con *peso* p_j per ogni vertice $j \in V$, indicando con $n := |V|$ il suo numero di vertici e con $m := |E|$ il suo numero di lati

Uno *Stable Set* (o *Indpendent Set*) di G è un sottinsieme di vertici $S \subseteq V$ tale che $E(S) = \emptyset$ (cioè nessun lato in E collega tra loro vertici di S)

Il problema dello *Stable Set* (o *Independent Set*, o *Vertex/Node Packing*) richiede di determinare uno Stable Set di G di peso *massimo*

Un modello ILP semplice per Stable Set, con m vincoli lineari, si ottiene utilizzando le variabili binarie:

$$x_j := \begin{cases} 1, & \text{se il vertice } j \text{ appartiene allo stable set} \\ 0, & \text{altrimenti} \end{cases}$$

ed ha la forma:

$$\max \sum_{j \in V} p_j x_j \tag{38}$$

$$x_i + x_j \leq 1, \quad (i,j) \in E \tag{39}$$

$$x_j \in \{0,1\}, \quad j \in V \tag{40}$$

La "debolezza" del corrispondente rilassamento continuo è evidente considerando il caso banale del problema in cui G è completo e, ad esempio, $p_j = 1$ per ogni $j \in V$:

- La soluzione ottima dell'ILP vale 1 (un solo vertice nello stable set)
- La soluzione ottima del rilassamento continuo è data da $x_j = 1/2$ per $j \in V$ e vale $n/2$

Un modello notevolmente più "forte" si ottiene dalla nozione di *Clique* di G, che corrisponde ad un sottinsieme di vertici $K \subseteq V$ tale che $E(K) = \{(i,j) : i,j \in K\}$ (cioè tutte le coppie di vertici di S hanno un lato in E che le collega)

Una Clique K si dice *massimale* se non esiste una clique K' tale che $K \subset K'$ (o, in altre parole, non esiste alcun vertice in $V \setminus K$ collegato a ciascun vertice di K da un lato in E)

Indicando con $\mathcal{K}$ la collezione delle clique massimali di G, ed osservando che ogni stable set può contenere al massimo un vertice di ciascuna clique, il modello più forte, con $|\mathcal{K}| = O(2^n)$ vincoli lineari, si ottiene sostituendo (39) con:

$$\sum_{j \in K} x_j \leq 1, \quad K \in \mathcal{K} \tag{41}$$

Più avanti nel corso verranno duscussi metodi per gestire il numero enorme di vincoli (41)

Il risultato che segue (senza dimostrazione) specifica il limite superiore al limite di sottoinsiemi in $\mathcal{K}$, che risulta essere esponenziale ma inferiore a 2^n:

Teorema 6 *Per un grafo di n vertici il numero massimo di clique massimali è pari a* $3^{n/3}$

Un esempio in cui il numero massimo è raggiunto si ottiene con $n = 3k$ per k intero ed

$E = \{(i,j) : i,j \in V, i \neq j\} \setminus \{(3i+1, 3i+2), (3i+1, 3i+3), (3i+2, 3i+3) : i = 1, \ldots, k\}$

in cui $\mathcal{K}$ è dato da tutti i sottoinsiemi di k vertici ottenuti selezionando esattamente un vertice tra $3i+1, 3i+2, 3i+3$ per $i = 1, \ldots, k$

Un modello "intermedio" tra i due visti sopra, con non più di m vincoli, come il primo, ma tutti "forti", come il secondo, si ottiene partendo dal primo modello e:

- sostituendo ciascun vincolo $x_i + x_j \leq 1$ con un vincolo della forma $\sum_{j \in K} x_j \leq 1$ per una qualche clique $K \in \mathcal{K}$ tale che $i, j \in K$ (osservando che è facile determinare una tale clique)
- eliminando eventuali vincoli ripetuti

Il corrispondente modello è definito da (38), (40) e:

$$\sum_{j \in K} x_j \leq 1, \quad \text{per } \textit{ogni } (i,j) \in E \text{ e per } \textit{una qualche } K \in \mathcal{K} \text{ tale che } i,j \in K \quad (42)$$

Esempio 6 Per il grafo $G = (V, E)$ con $V = \{1,2,3,4,5,6\}$ ed $E = \{(1,2), (1,3), (1,4), (1,5), (1,6), (2,5), (2,6), (5,6)\}$ i vincoli (39) hanno la forma:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 + x_3 &\leq 1 \\ x_1 + x_4 &\leq 1 \\ x_1 + x_5 &\leq 1 \\ x_1 + x_6 &\leq 1 \\ x_2 + x_5 &\leq 1 \\ x_2 + x_6 &\leq 1 \\ x_5 + x_6 &\leq 1 \end{aligned}$$

mentre $\mathcal{K} = \{\{1,2,5,6\}, \{1,3\}, \{1,4\}\}$ e quindi i vincoli (41) hanno la forma:

$$\begin{aligned} x_1 + x_2 + x_5 + x_6 &\leq 1 \\ x_1 + x_3 &\leq 1 \\ x_1 + x_4 &\leq 1 \end{aligned}$$

Infine, in questo semplice caso i vincoli (42) coincidono con i vincoli (41)

Stable Set e Set Packing

Viene ora mostrato come il problema dello Stable Set ed il problema del Set Packing definito da (22) in realtà coincidano

Innanzitutto, si osservi come tutti e tre i modelli ILP mostrati per Stable Set siano di tipo Set Packing

Viceversa:

Osservazione 3 *Dato il problema del Set Packing (22), associato alla matrice $A \in \{0,1\}^{m \times n}$ ed al vettore di costi c, esso è equivalente al problema di Stable Set associato al grafo $G(A) = (V, E)$ con vertici $V := \{1, \ldots, n\}$, pesi $p_j := c_j$ per $j \in V$, e lati:*

$$E := \{(i,j) : a_{hi} = a_{hj} = 1 \text{ per qualche } h \in \{1, \ldots, m\}\}$$

Dim. Due variabili x_j e x_j possono valere entrambe 1 in una soluzione di Set Packing se e solo se non esiste un vincolo h per cui $a_{hi} = a_{hj} = 1$, cioè se e solo se i vertici i e j in $G(A)$ non sono collegati da un lato

Questo implica che ciascuna soluzione di Set Packing corrisponde ad uno Stable Set di G(A) e viceversa □

La visualizzazione del problema del Set Packing su grafo è utile per verificare se le corrispondenti disuguaglianze siano "forti", e per "rafforzarle" in caso contrario

Esempio 7 Si consideri il seguente problema di Set Packing:

$$\begin{array}{rcl} \max \sum_{j=1}^{6} c_j x_j & & \\ x_1 + x_4 + x_6 & \leq & 1 \\ x_2 + x_4 + x_5 & \leq & 1 \\ x_3 + x_4 & \leq & 1 \\ x_2 + x_3 + x_5 & \leq & 1 \\ x_j & \in & \{0,1\}, \quad j = 1, \ldots, 6 \end{array}$$

In questo caso, $G(A) = (V, E)$ con $V = \{1,2,3,4,5,6\}$ ed $E = \{(1,4),(1,6),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6)\}$

Il primo vincolo corrisponde alla clique massimale $\{1,4,6\}$ di $G(A)$, mentre i tre vincoli successivi possono essere sostituiti dalla seguente disuguaglianza corrispondente alla clique massimale $\{2,3,4,5\}$ di $G(A)$, "rafforzando" il modello:

$$x_2 + x_3 + x_4 + x_5 \leq 1$$

Disuguaglianze di tipo Clique e (M)ILP

Più in generale, il seguente procedimento è fondamentale per il "rafforzamento" di disuguaglianze della forma $\sum_{j \in S} x_j \leq 1$ in cui x_j è una variabile binaria per $j \in S$, contenute in un generico ILP o MILP (in cui altre variabili possono essere continue e/o intere ma non binarie, e altri vincoli possono avere una forma qualsiasi):

- Definire il grafo $G = (V, E)$ in cui V corrisponde all'insieme delle variabili binarie ed i vertici corrispondenti a due variabili binarie sono collegati da un lato in E se le due variabili *non* valgono entrambe 1 in alcuna soluzione del (M)ILP
- Per ogni disuguaglianza nella forma sopra, verificare se S è una clique massimale di G e, in caso contrario, determinare una clique massimale S' tale che $S \subset S'$ e sostituire la disuguaglianza con $\sum_{j \in S'} x_j \leq 1$

Vertex Coloring

Dato un grafo non orientato $G = (V, E)$ con $n := |V|$ vertici ed $m := |E|$ lati, il problema del *Vertex Coloring* richiede di assegnare dei *colori* ai vertici di G in modo che:

- vertici collegati da un lato in E ricevano colori *diversi*
- il numero di colori utilizzati sia minimizzato

Ispirandosi al primo modello visto per il Bin Packing, osservando che n colori sono sempre sufficienti (e necessari se e solo se il grafo è completo), ed introducendo le variabili binarie:

$$y_j := \begin{cases} 1, & \text{se il colore } j \text{ è utilizzato} \\ 0, & \text{altrimenti} \end{cases}$$

$$x_{ij} := \begin{cases} 1, & \text{se al vertice } i \text{ è assegnato il colore } j \\ 0, & \text{altrimenti} \end{cases}$$

il modello più semplice per il problema ha la forma:

$$\min \sum_{j=1}^{n} y_j \tag{43}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in V \tag{44}$$

$$x_{ij} + x_{hj} \leq y_j, \quad (i,h) \in E,\ j = 1, \ldots, n \tag{45}$$

$$y_j, x_{ij} \in \{0,1\}, \quad i \in V,\ j = 1, \ldots, n \tag{46}$$

Lo svantaggio di questo modello combina gli svantaggi del modello analogo per il Bin Packing e del primo modello visto per lo Stable Set, come mostrato ad esempio dal caso in cui G è completo, per cui:

- La soluzione ottima dell'ILP vale n (un colore diverso per ogni vertice)
- La soluzione ottima del rilassamento continuo è data da $y_1 = y_2 = 1$; $x_{i1} = x_{i2} = 1/2$ per $i \in V$, e vale 2

In analogia al problema dello Stable Set, il modello può essere "rafforzato" sostituendo i vincoli (45) con i vincoli:

$$\sum_{i \in K} x_{ij} \quad \leq \quad y_j, \quad K \in \mathcal{K},\ j = 1, \ldots, n \tag{47}$$

dove $\mathcal{K}$ indica la collezione di clique massimali di G

Rimangono in ogni caso gli svantaggi legati al modello analogo per il Bin Packing (a cui si aggiunge il fatto di avere $O(n2^n)$ disuguaglianze)

Ad esempio, il valore ottimo della soluzione del rilassamento continuo è pari alla *massima cardinalità* di una clique in G, che corrisponde ad un lower bound ovvio sul numero di colori necessari

In analogia al Bin Packing, un modello ILP alternativo di gran lunga più utile in pratica si ottiene osservando che l'insieme dei vertici che ricevono lo stesso colore in una soluzione del Vertex Coloring corrisponde ad uno Stable Set

Considerando la collezione $\mathcal{S}$ degli Stable Set *massimali* di G ed introducendo una variabile binaria per ciascuno di essi:

$$x_S := \begin{cases} 1, & \text{se tutti e soli i vertici in } S \text{ ricevono lo stesso colore in soluzione} \\ 0, & \text{altrimenti} \end{cases}$$

un modello di tipo Set Covering analogo al modello (37) per il Bin Packing è il seguente:

$$\begin{aligned} \min & \sum_{S \in \mathcal{S}} x_S \\ & \sum_{S \in \mathcal{S}: i \in S} x_S \quad \geq \quad 1, \quad i \in V \\ & x_S \quad \in \quad \{0,1\}, \quad S \in \mathcal{S} \end{aligned} \tag{48}$$

(Per l'analogo modello di tipo Set Partitioning, con una variabile per ogni Stable Set – non necessariamente massimale – di G, valgono le stesse considerazioni fatte per il Bin Packing)

Esempio 8 Per il grafo $G = (V, E)$ con $V = \{1, 2, 3, 4, 5, 6\}$ ed $E = \{(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 5), (2, 6), (5, 6)\}$ si ha $\mathcal{S} = \{\{1\}, \{2, 3, 4\}, \{3, 4, 5\}, \{3, 4, 6\}\}$, ed il corrispondente modello (48) ha la forma:

$$\begin{array}{rcl} \min \sum_{S \in \mathcal{S}} x_S & & \\ x_{\{1\}} & \geq & 1 \\ x_{\{2,3,4\}} & \geq & 1 \\ x_{\{2,3,4\}} + x_{\{3,4,5\}} + x_{\{3,4,6\}} & \geq & 1 \\ x_{\{2,3,4\}} + x_{\{3,4,5\}} + x_{\{3,4,6\}} & \geq & 1 \\ x_{\{3,4,5\}} & \geq & 1 \\ x_{\{3,4,6\}} & \geq & 1 \\ x_S & \in & \{0, 1\}, \quad S \in \mathcal{S} \end{array}$$

Si osservi infine che il problema duale del rilassamento continuo di (48) ha la forma:

$$\begin{array}{rcll} \max \sum_{i \in V} y_i & & & \\ \sum_{i \in S} y_i & \leq & 1, \quad S \in \mathcal{S} & (49) \\ y_i & \geq & 0, \quad i \in V & \end{array}$$

e corrisponde al rilassamento continuo del modello per il problema di determinare la *Clique* di peso massimo in G (con tutti i pesi pari ad 1) analogo al modello (38), (41), (40) per il problema di determinare lo Stable Set di peso massimo

Più precisamente, il problema della Clique su di un grafo $G = (V, E)$ coincide con il problema dello Stable Set sul grafo $\overline{G} = (V, \overline{E})$, detto grafo *complemento* di G, che ha lo stesso insieme di vertici di G e contiene tutti e soli i lati che G non ha:

$$\overline{E} := \{(i, j) : i, j \in V, i \neq j, (i, j) \notin E\}$$

Quest'ultima osservazione permette di stabilire un limite superiore al numero di elementi in $\mathcal{S}$, dato che il Teorema 6 implica che il numero massimo di insiemi stabili massimali di un grafo con n vertici è pari a $3^{n/3}$

Traveling Salesman

Il celebre problema del *Traveling Salesman* (TSP) è generalmente definito su grafo non orientato

Dato che il modello ILP per la versione su grafo orientato, detta *Asymmetric TSP* (ATSP), è più generale e semplice, definiremo prima quest'ultimo

Dato un grafo orientato $G = (V, A)$, completo e con costo $c_a = c_{(i,j)}$ per ogni arco $a = (i, j) \in A$, l'ATSP richiede di determinare un circuito di G che:

- visiti *una ed una sola* volta ciascun vertice $i \in V$
- abbia costo minimo, dove per costo del circuito si intende la somma dei costi degli archi nel circuito

Si osservi che la richiesta "ed una sola volta", che sembra essere poco realistica in problemi di percorsi ottimi (se conviene, è naturale passare anche più volte da uno stesso punto), risulta essere ridondante qualora i costi soddisfino la *disuguaglianza triangolare*

$$c_{(i,j)} + c_{(j,k)} \geq c_{(i,k)}, \quad i, j, k \in V, i \neq j, i \neq k, j \neq k \tag{50}$$

come spesso avviene in pratica (si rimanda alla parte delle applicazioni, problema del Vehicle Routing, per delucidazioni a riguardo)

Per l'ATSP (così come per il TSP) esiste un unico modello ILP che viene usato con successo nella pratica, che utilizza le variabili binarie:

$$x_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene al circuito} \\ 0, & \text{altrimenti} \end{cases}$$

Osservando che ogni circuito che visita tutti i vertici una ed una sola volta ha esattamente un arco entrante ed un arco uscente in ciascun vertice di G, si potrebbe pensare che il seguente modello sia corretto:

$$\min \sum_{a \in A} c_a x_a \tag{51}$$

$$\sum_{a \in \delta^-(i)} x_a = 1, \quad i \in V \tag{52}$$

$$\sum_{a \in \delta^+(i)} x_a = 1, \quad i \in V \tag{53}$$

$$x_a \in \{0, 1\}, \quad a \in A \tag{54}$$

Esempio 9 Per un grafo di 6 vertici i vincoli (52) e (53) hanno la forma:

$$\begin{aligned} x_{(2,1)} + x_{(3,1)} + x_{(4,1)} + x_{(5,1)} + x_{(6,1)} &= 1 \\ &\dots \\ x_{(1,6)} + x_{(2,6)} + x_{(3,6)} + x_{(4,6)} + x_{(5,6)} &= 1 \\ x_{(1,2)} + x_{(1,3)} + x_{(1,4)} + x_{(1,5)} + x_{(1,6)} &= 1 \\ &\dots \\ x_{(6,1)} + x_{(6,2)} + x_{(6,3)} + x_{(6,4)} + x_{(6,5)} &= 1 \end{aligned}$$

In realtà, una soluzione di questo modello, corrispondente ad un sottoinsieme di archi che, per ogni vertice, contiene esattamente un arco entrante ed un arco uscente, può corrispondere a *più di un circuito*

Il modello sopra non è quindi completo per l'ATSP, e corrisponde in realtà al modello ILP per il problema polinomiale dell'*Assegnamento*, il cui rilassamento continuo definisce la chiusura convessa delle soluzioni intere ammissibili

Per ottenere un modello per l'ATSP occorre aggiungere dei vincoli che proibiscano i *sottocircuiti*, corrispondenti a circuiti che visitano solo un sottoinsieme di vertici

Indicando con $\mathcal{C}$ la collezione di tutti i sottocircuiti di G, un insieme di vincoli che, aggiunti al modello sopra, forniscono un modello ILP per l'ATSP è:

$$\sum_{a \in C} x_a \quad \leq \quad |C| - 1, \quad C \in \mathcal{C} \tag{55}$$

Si osservi che il numero di sottocircuiti di k archi (che cioè visitano k vertici) di un grafo con n vertici è pari a $\binom{n}{k}(k-1)!$, in quanto ci sono $\binom{n}{k}$ modi di scegliere i k vertici visitati e, per ogni scelta, $(k-1)!$ modi di definire un sottocircuito che visiti tali vertici

Esempio 10 Per un grafo di 6 vertici i vincoli (55) hanno la forma:

$$\begin{aligned}
x_{(1,2)} + x_{(2,1)} &\leq 1 \\
x_{(1,3)} + x_{(3,1)} &\leq 1 \\
&\dots \\
x_{(1,2)} + x_{(2,3)} + x_{(3,1)} &\leq 2 \\
x_{(1,3)} + x_{(3,2)} + x_{(2,1)} &\leq 2 \\
&\dots \\
x_{(1,2)} + x_{(2,3)} + x_{(3,4)} + x_{(4,1)} &\leq 3 \\
x_{(1,2)} + x_{(2,4)} + x_{(4,3)} + x_{(3,1)} &\leq 3 \\
x_{(1,3)} + x_{(3,2)} + x_{(2,4)} + x_{(4,1)} &\leq 3 \\
x_{(1,3)} + x_{(3,4)} + x_{(4,2)} + x_{(2,1)} &\leq 3 \\
x_{(1,4)} + x_{(4,2)} + x_{(2,3)} + x_{(3,1)} &\leq 3 \\
x_{(1,4)} + x_{(4,3)} + x_{(3,2)} + x_{(2,1)} &\leq 3 \\
&\dots
\end{aligned}$$

Esiste una versione molto migliore dei vincoli (55), che è quella che viene utilizzata nella pratica in quanto contiene meno vincoli più forti

Dato un sottoinsieme $S \subseteq V$ ed un sottocircuito che visita i vertici in S, i vincoli (55) specificano che al massimo $|S| - 1$ archi *nel sottocircuito* possono essere selezionati in soluzione: *in realtà* è facile verificare che al massimo $|S| - 1$ archi *che collegano vertici di* S possono essere selezionati in soluzione

Quest'ultima osservazione porta ai seguenti vincoli più forti di eliminazione di sottocircuiti, noti come *Subtour Elimination Constraints*:

$$\sum_{a \in A(S)} x_a \quad \leq \quad |S| - 1, \quad S \subseteq V, 2 \leq |S| \leq |V| - 2 \tag{56}$$

Si osservi che non è necessario introdurre i vincoli (56) quando $|S| = 1$ e $|S| = |V| - 1$ in quanto nessuna soluzione del modello (51)-(54) può contenere un sottocircuito che visita un solo vertice

Per ogni sottoinsieme $S \subseteq V$ tale che $2 \leq |S| \leq |V| - 2$, l'*unico* vincolo (56) associato ad S *domina* gli $(|S| - 1)!$ vincoli (55) associati ad S

Il modello ILP per l'ATSP è quindi dato da (51)-(54) con l'aggiunta dei $2^n - 2(n + 1)$ vincoli (56)

Esempio 11 Per un grafo di 6 vertici i vincoli (56) che sostituiscono i vincoli (55) riportati esplicitamente nell'Esempio 10 hanno la forma:

$$\begin{aligned}
x_{(1,2)} + x_{(2,1)} &\leq 1 \\
x_{(1,3)} + x_{(3,1)} &\leq 1 \\
&\cdots \\
x_{(1,2)} + x_{(1,3)} + x_{(2,1)} + x_{(2,3)} + x_{(3,1)} + x_{(3,2)} &\leq 2 \\
&\cdots \\
x_{(1,2)} + x_{(1,3)} + x_{(1,4)} + x_{(2,1)} + x_{(2,3)} + x_{(2,4)} + x_{(3,1)} + x_{(3,2)} + x_{(3,4)} + x_{(4,1)} + x_{(4,2)} + x_{(4,3)} &\leq 3 \\
&\cdots
\end{aligned}$$

Esiste un modo alternativo ed equivalente di esprimere i vincoli (56):

$$\sum_{a \in \delta^+(S)} x_a \quad \geq \quad 1, \quad S \subseteq V, 2 \leq |S| \leq |V| - 2 \tag{57}$$

L'equivalenza segue da:

Osservazione 4 *Dato un vettore $x = (x_a)$ che soddisfa (52) e (53), x soddisfa (56) se e solo se soddisfa (57)*

Dim. Si consideri x che soddisfa (52) e (53) assieme ad un generico sottoinsieme $S \subseteq V$, tale che $2 \leq |S| \leq |V| - 2$

Innanzitutto vale l'identità:

$$\sum_{i \in S} \sum_{a \in \delta^+(i)} x_a = \sum_{a \in A(S)} x_a + \sum_{a \in \delta^+(S)} x_a$$

Inoltre, poiché x soddisfa (53), si ha che:

$$\sum_{i \in S} \sum_{a \in \delta^+(i)} x_a = |S|$$

Combinando le due equazioni sopra si ottiene:

$$\sum_{a \in A(S)} x_a + \sum_{a \in \delta^+(S)} x_a = |S|$$

che implica:

$$\sum_{a \in A(S)} x_a \leq |S| - 1 \Leftrightarrow \sum_{a \in \delta^+(S)} x_a \geq 1$$

ovverosia x soddisfa il vincolo (56) per S se e solo se soddisfa il vincolo (57) per S $\square$

Esempio 12 Per un grafo di 6 vertici i vincoli (57) corrispondenti ai vincoli (56) riportati esplicitamente nell'Esempio 11 hanno la forma:

$$\begin{aligned}
x_{(1,3)} + x_{(1,4)} + x_{(1,5)} + x_{(1,6)} + x_{(2,3)} + x_{(2,4)} + x_{(2,5)} + x_{(2,6)} &\geq 1 \\
x_{(1,2)} + x_{(1,4)} + x_{(1,5)} + x_{(1,6)} + x_{(3,2)} + x_{(3,4)} + x_{(3,5)} + x_{(3,6)} &\geq 1 \\
&\dots \\
x_{(1,4)} + x_{(1,5)} + x_{(1,6)} + x_{(2,4)} + x_{(2,5)} + x_{(2,6)} + x_{(3,4)} + x_{(3,5)} + x_{(3,6)} &\geq 1 \\
&\dots \\
x_{(1,5)} + x_{(1,6)} + x_{(2,5)} + x_{(2,6)} + x_{(3,5)} + x_{(3,6)} + x_{(4,5)} + x_{(4,6)} &\geq 1 \\
&\dots
\end{aligned}$$

Si osservi che l'equivalenza tra (56) e (57) vale a seguito degli altri vincoli, e che ci sono problemi in cui si cerca un circuito che non visiti necessariamente tutti i vertici per i quali solo uno dei due tra (56) e (57) (opportunamente adattati al caso specifico) è corretto – si vedranno esempi nel seguito

Il TSP è la variante dell'ATSP definita su grafo non orientato, ovverosia il problema, dato un grafo non orientato $G = (V, E)$, completo e con costo $c_e = c_{(i,j)}$ per ogni lato $e = (i, j) \in E$, di determinare un circuito di costo minimo che visiti una ed una sola volta ciascun vertice

Chiaramente, il TSP è il caso particolare dell'ATSP in cui $c_{(i,j)} = c_{(j,i)}$ per $i, j \in V, i \neq j$, e quindi i modelli visti per l'ATSP sono utilizzabili anche per il TSP

D'altra parte, TSP viene generalmente risolto tramite il modello ILP in cui le variabili sono associate ai lati di G:

$$x_e := \begin{cases} 1, & \text{se il lato } e \text{ appartiene al circuito} \\ 0, & \text{altrimenti} \end{cases}$$

e che ha la forma:

$$\min \sum_{e \in E} c_e x_e \tag{58}$$

$$\sum_{e \in \delta(i)} x_e = 2, \quad i \in V \tag{59}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad S \subseteq V, 2 \leq |S| \leq |V| - 2 \tag{60}$$

$$x_e \in \{0, 1\}, \quad e \in E \tag{61}$$

Si osservi che, nel caso di $|S| = 2$, il vincolo (60) associato ad $S = \{i, j\}$ assume la forma $x_{(i,j)} \leq 1$ – questi vincoli sono implicati ovviamente da (61), ma sono necessari nel rilassamento continuo

In alternativa ai vincoli (60), ed equivalenti come nel caso dell'ATSP, si possono usare i seguenti vincoli:

$$\sum_{e \in \delta(S)} x_e \geq 2, \quad S \subseteq V, 2 \leq |S| \leq |V| - 2 \tag{62}$$

Sommario dei Problemi e Modelli Illustrati

La seguente tabella elenca i problemi ed i corrispondenti modelli ILP illustrati precedentemente:

Problema	modello ILP	(#variabili, #vincoli)	Il corrispondente rilassamento continuo è
UFLP	(11), (12), (16), (14)	$(n + mn,\ m + n)$	"debole"
	(11), (12), (15), (14)	$(n + mn,\ m + mn)$	"forte"
Set Covering	(20)	$(n,\ m)$	"forte"
Set Partionining	(21)	$(n,\ m)$	"forte"
Set Packing	(22)	$(n,\ m)$	"intermedio" purchè tutti i vincoli corrispondano a clique massimali di $G(A)$ (vedi Stable Set)
CFLP	(23)-(27)	$(n + mn,\ m + mn + n)$	generalmente "forte", anche se tende a diventare "debole" quando gli impianti tendono a essere uguali (vedi Bin Packing)
Bin Packing	(28)-(32)	$(n + mn,\ m + mn + n)$	"debole"
	(37)	$(O(2^n),\ n)$	"forte"
Stable Set	(38)-(40)	$(n,\ m)$	"debole"
	(38), (41), (40)	$(n,\ O(2^n))$	"forte"
	(38), (42), (40)	$(n,\ O(m))$	"intermedio"
Vertex Coloring	(43)-(46)	$(n + n^2,\ n + nm)$	"debole"
	(43), (44), (47), (46)	$(n + n^2,\ n + O(n2^n))$	"intermedio"
	(48)	$(O(2^n),\ n)$	"forte"
ATSP	(51)-(54), (56)	$(n(n-1),\ 2n + O(2^n))$	"forte"
TSP	(58)-(61)	$(n(n-1)/2,\ n + O(2^n))$	"forte"

Metodi di Soluzione di LP e (M)ILP

In questo corso viene posta poca enfasi sui metodi risolutivi per LP e (M)ILP in quanto, nella pratica, questi problemi vengono risolti tramite l'uso di risolutori commerciali o public domain

In ogni caso, facendo in parte riferimento a quanto illustrato nei corsi precedenti, verranno brevemente illustrati i metodi principalmente utilizzati da tali risolutori

Un caso estremamente rilevante in cui *non è possibile* fornire ad un risolutore un modello (MI)LP ed aspettare che lo risolva è quello in cui il numero di vincoli e/o variabili sia troppo elevato (ad esempio esponenziale nella dimensione dell'input del problema rappresentato)

Verranno quindi illustrati in un certo dettaglio i metodi per la soluzione di LP con moltissimi vincoli e/o variabili, che a loro volta sono risolti come rilassamenti continui di (M)ILP

Algoritmi per LP

Esistono due metodi utilizzati principalmente nella pratica per la soluzione di LP:

- *Algoritmo del Simplesso*, che è un algoritmo che esplora i vertici della regione ammissibile (un *poliedro*), ed è quindi di tipo "combinatorio" in quanto il numero di vertici è finito
- *Metodi a Punto Interno*, che si muovono all'interno della regione ammissibile e sono basati su tecniche più generali di Programmazione Nonlineare

Fino all'inizio degli anni '90, i Metodi a Punto Interno non erano competitivi con l'Algoritmo del Simplesso, mentre attualmente tendono ad essere migliori quando si tratta di risolvere *un unico LP* di grandi dimensioni (ad esempio con 10^7 variabili e 10^4 vincoli)

Il vantaggio *enorme* attuale dell'Algoritmo del Simplesso è dato dalla possibilità di *riottimizzare velocemente* un LP precedentemente risolto ed al quale siano stati aggiunti dei vincoli o delle variabili, come spesso accade (si veda il seguito)

Più precisamente, esistono due versioni dell'Algoritmo del Simplesso:

- Algoritmo del Simplesso *Primale*, che parte da un vertice della regione ammissibile, e, ad ogni iterazione, verifica l'esistenza di vertici *adiacenti* a quello corrente di valore migliore, spostandosi ad uno di tali vertici se ne esisitono e fermandosi altrimenti poiché il vertice corrente è ottimo
- Algoritmo del Simplesso *Duale*, che opera come il Primale sul problema duale (anche se l'algoritmo lavora esplicitamente con il problema primale e solo implicitamente con il duale)

Si supponga di avere risolto un LP, ottenendo le soluzioni ottime primale x^* e duale y^* (tutti i risolutori le forniscono entrambe):

- Se vengono aggiunti uno o più vincoli, la soluzione y^* rimane ammissibile per il duale (ponendo le variabili duali corrispondenti ai nuovi vincoli aggiunti a 0), e quindi il nuovo LP può essere risolto tramite l'Algoritmo del Simplesso Duale ripartendo da y^*
- Se vengono aggiunte una o più variabili, la soluzione x^* rimane ammissibile per il primale (ponendo le nuove variabili primali a 0), e quindi il nuovo LP può essere risolto tramite l'Algoritmo del Simplesso Primale ripartendo da x^*

Fino a quando i Metodi a Punto Interno non saranno competitivi con l'Algoritmo del Simplesso nella fase di riottimizzazione, quest'ultimo rimarrà indiscutibilmente il metodo più utilizzato per risolvere LP (e quindi (M)ILP)

Algoritmi per (M)ILP

Il metodo di gran lunga più utilizzato per la soluzione di (M)ILP, che è quello applicato dai risolutori, è il *Metodo Branch-and-Bound*

Si consideri un generico problema ILP di *minimo* (per un problema di *massimo* il procedimento è del tutto analogo)

Il metodo genera dei problemi da altri per cui lavora con una *lista* di problemi da considerare

Si indichi inoltre con $c^T x$ la funzione obiettivo, e con z il valore della soluzione ottima (per semplicità non si riporta la memorizzazione della soluzione ottima, che si effettua in modo ovvio)

1. Inizializza la lista con il problema ILP da risolvere e poni $z := +\infty$
2. Se la lista è vuota *STOP* (z è il valore della soluzione ottima), altrimenti seleziona un problema P dalla lista
3. Risolvi il rilassamento continuo di P ottenendo la soluzione x^*
4. Se x^* è intera allora poni $z := \min\{z, c^T x^*\}$ e vai a 2
5. Se $c^T x^* \geq z$ allora vai a 2 (il valore della soluzione ottima di P non può essere migliore di z)
6. Considera una variabile x_j tale che x_j^* è frazionario, genera due problemi, il primo ottenuto da P aggiungendo il vincolo $x_j \leq \lfloor x_j^* \rfloor$ ed il secondo ottenuto da P aggiungendo il vincolo $x_j \geq \lceil x_j^* \rceil$, aggiungi i due problemi generati alla lista e vai a 2

(Il metodo si adatta in maniera ovvia a MILP)

Si noti che lo schema sopra lascia due gradi fondamentali di libertà: quale problema selezionare al passo 2 e quale variabile considerare al passo 6 (scelte differenti hanno effetti differenti, non discussi in questa sede)

Per ogni nuovo problema generato, il rilassamento continuo viene risolto con l'Algoritmo del Simplesso Duale a partire dalla soluzione del problema generante

In ogni caso, praticamente tutto il tempo di esecuzione è richiesto dalla soluzione di LP

Per evitare la crescita smisurata della lista, è necessario che i test nei passi 4 e 5 siano spesso verificati – questo richiede che il rilassamento continuo sia sufficientemente "vicino" alla chiusura convessa delle soluzioni dell'ILP

In ogni caso, per alcuni problemi NP-completi, anche i modelli ILP più "forti" non riescono ad evitare la crescita smisurata della lista, ed occorre ripiegare su *algoritmi euristici* per determinare soluzioni "buone" anche se non necesariamente ottime

Algoritmi Euristici per (M)ILP

Il campo degli algoritmi euristici è estremamente vasto dato che per molti problemi di interesse pratico non è possibile determinare la soluzione ottima

In molti casi, la realizzazione di un algoritmo euristico efficiente richiede un notevole sforzo di programmazione

D'altra parte, la struttura fondamentale di molti tra i migliori algoritmi euristici può essere riprodotta con sforzo di programmazione quasi nullo, seguendo lo schema seguente per un generico ILP (che nuovamente si adatta in maniera ovvia a MILP):

1. Poni P uguale al problema ILP da risolvere
2. Risolvi il rilassamento continuo di P ottenendo la soluzione x^*
3. Se x^* è intera allora $STOP$ (x^* è la soluzione euristica cercata)
4. Fissa alcune delle variabili x_j ad un valore intero "suggerito" da x^*, aggiungendo le corrispondenti equazioni $x_j = a$ a P e vai a 2

Ovviamente a seconda della scelta delle variabili da fissare al passo 4 si ottengono algoritmi e soluzioni che possono essere molto diversi

Un esempio di passo 4 per il problema del Set Covering è il seguente:

4. Aggiungi a P il vincolo $x_j = 1$ per ogni j tale che $x_j^* = 1$ e per j tale che $x_j^* = \max\{x_k^* : x_k^* < 1\}$

Il corso di Ottimizzazione delle Risorse LS fornisce una descrizione approfondita dei principali metodi di definizione di algoritmi euristici ponendo l'enfasi su applicazioni pratiche

Modelli con Molti Vincoli e Separazione

I metodi illustrati sopra richiedono la soluzione di LP – come detto questa non può essere ottenuta semplicemente fornendo l'LP corrispondente ad un risolutore se questo ha una dimensione eccessiva

Verranno considerati inizialmente LP il cui numero di vincoli è troppo elevato, quali ad esempio il rilassamento continuo dei modelli (51)-(54), (56) per l'ATSP e (38), (41), (40) per lo Stable Set

In generale, si consideri il generico problema LP:

$$\begin{array}{rcl} \min c^T x & & \\ Ax & \geq & b \\ x & \geq & 0 \end{array} \tag{63}$$

in cui il numero di disuguaglianze è molto grande

Il metodo per risolverlo senza dovere fornire tutti i vincoli al risolutore è il seguente:

1. Inizializza $\tilde{A}$, $\tilde{b}$ con un ("piccolo") sottoinsieme delle righe in A, b
2. Risolvi l'LP con un sottoinsieme di vincoli rispetto a (63):

$$\begin{array}{rcl} \min c^T x & & \\ \tilde{A}x & \geq & \tilde{b} \\ x & \geq & 0 \end{array} \tag{64}$$

   ottenendo la soluzione x^*
3. Se x^* soddisfa tutti i vincoli in $Ax \geq b$ allora *STOP* (x^* è la soluzione ottima di (63)), altrimenti aggiungi le righe corrispondenti ad alcuni dei vincoli violati a $\tilde{A}$, $\tilde{b}$ e vai a 2

Si osservi che lo schema è molto semplice e diretto, dato che aggiunge vincoli solo quando "servono", cioè quando sono violati

Nonostante la sua semplicità il procedimento funziona piuttosto bene in pratica, nel senso che il numero finale di vincoli in $\tilde{A}x \geq \tilde{b}$ tende ad essere enormemente più piccolo del numero totale di vincoli, come motivato anche da alcuni risultati teorici piuttosto complessi e qui non enunciati

I gradi di libertà consistono nella determinazione dell'insieme iniziale di vincoli e nella scelta di quali vincoli violati tra quelli individuati aggiungere

Da un punto di vista pratico le scelte corrispondenti ai due gradi di libertà non risultano essere "critiche", nel senso che il procedimento tende a convergere velocemente per qualunque

insieme iniziale di vincoli (purchè ovviamente l'LP iniziale sia *limitato*) ed aggiungendo un solo vincolo violato alla volta

La parte delicata è invece il test al passo 3 per l'individuazione di eventuali vincoli violati, detto *Separazione*

In particolare, se il numero totale di vincoli è troppo elevato, *non solo* questi vincoli non possono essere forniti tutti ad un risolutore, *ma anche* non possono essere controllati esplicitamente uno ad uno

L'alternativa al controllo di tutti i vincoli uno ad uno è la definizione e soluzione di un opportuno problema, spesso tramite ILP, come illustrato nel seguito per alcuni esempi

(Il fatto di dovere risolvere un ILP all'interno di una procedura per la soluzione di un LP che a sua volta risulta spesso essere il rilassamento continuo di un ILP può sembrare estremamente inefficiente, ma la realtà è che anche risolvendo un ILP per la separazione il procedimento funziona bene in pratica)

Si osservi che, durante la procedura iterativa di aggiunta dei vincoli e soluzione dell'LP ridotto, il valore ottimo $c^T x^*$ è sempre *inferiore* a quello dell'LP completo, e risulta essere quindi un lower bound valido sul valore ottimo dell'eventuale ILP di cui l'LP completo è il rilassamento continuo

In base all'osservazione sopra, anche se la procedura viene interrotta (ad esempio perché si usa un algoritmo euristico per risolvere il problema di separazione e questo non trova vincoli violati) un lower bound è disponibile in ogni caso

ATSP

Come primo esempio consideriamo la soluzione del rilassamento continuo del modello (51)-(54), (56) per l'ATSP

Al passo 1, è naturale inserire nell'LP iniziale le sole equazioni (52) e (53)

Il problema della separazione da risolvere al passo 3 si riferisce quindi alla ricerca di vincoli (56) violati da x^*, e può essere formulato come segue:

Dato $x^* = (x^*_a)$, determinare, se esiste, un sottoinsieme $S^* \subseteq V$ tale che:

- $\sum_{a \in A(S^*)} x^*_a > |S^*| - 1 \Leftrightarrow |S^*| - \sum_{a \in A(S^*)} x^*_a < 1$
- $2 \leq |S^*| \leq |V| - 2$

Anzichè considerare esplicitamente tutti i possibili sottoinsiemi S^* (impraticabile per valori di $|V|$ non troppo piccoli), il problema di separazione può essere risolto introducendo le variabili binarie:

$$y_i := \begin{cases} 1, & \text{se il vertice } i \text{ appartiene a } S^* \\ 0, & \text{altrimenti} \end{cases}$$

e

$$z_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene a } A(S^*) \\ 0, & \text{altrimenti} \end{cases}$$

e determinando, se esiste, una soluzione al seguente sistema:

$$\sum_{i \in V} y_i - \sum_{a \in A} x_a^* z_a < 1 \tag{65}$$

$$z_{(i,j)} = 1 \Leftrightarrow y_i = y_j = 1, \quad (i,j) \in A \tag{66}$$

$$\sum_{i \in V} y_i \geq 2 \tag{67}$$

$$\sum_{i \in V} y_i \leq |V| - 2 \tag{68}$$

$$y_i, z_a \in \{0,1\}, \quad i \in V,\ a \in A \tag{69}$$

(Sarebbe in realtà facile mostrare che i vincoli (67) e (68) possono essere eliminati)

Si noti che, nel problema di separazione, i valori x_a^* sono ovviamente coefficienti noti

Il vincolo logico (67) può essere espresso dalle seguenti tre disuguaglianze lineari:

$$z_{(i,j)} \leq y_i, \quad (i,j) \in A \tag{70}$$

$$z_{(i,j)} \leq y_j, \quad (i,j) \in A \tag{71}$$

$$z_{(i,j)} \geq y_i + y_j - 1, \quad (i,j) \in A \tag{72}$$

Inoltre, la "strana" disuguaglianza con minore stretto (65) può essere sostituita dalla funzione obiettivo:

$$\min \sum_{i \in V} y_i - \sum_{a \in A} x_a^* z_a \tag{73}$$

Risolvendo il corrispondente ILP (73), (67)-(69), (70)-(72), se la soluzione ottima $(\overline{y}, \overline{z})$ ha valore ≥ 1, allora *tutti* i vincoli (56) sono soddisfatti da x^*, altrimenti l'insieme $S^* := \{i \in V : \overline{y}_i = 1\}$ definisce il vincolo (56) *più violato* da x^*

Esempio 13 Si consideri il grafo completo con 9 vertici e costi 1 per tutti gli archi ad eccezione di quelli in $A(\{1,2,3\}) \cup A(\{4,5,6\}) \cup A(\{7,8,9\}) \cup A(\{3,6,9\})$, che hanno costo 0

Imponendo inizialmente in $\tilde{A}x \geq \tilde{b}$ i 18 vincoli (52) e (53), il procedimento sopra esegue le seguenti iterazioni:

1. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,3),(3,1),(4,5),(5,6),(6,4),(7,8),(8,9),(9,7)\}$$

ed il vincolo (56) associato a $S^* = \{1,2,3\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

2. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,1),(3,6),(6,9),(9,3),(4,5),(5,4),(7,8),(8,7)\}$$

ed il vincolo (56) associato a $S^* = \{1,2\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

3. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,3),(3,9),(9,8),(8,7),(7,1),(4,5),(5,6),(6,4)\}$$

ed il vincolo (56) associato a $S^* = \{4,5,6\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

4. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,3),(3,6),(6,9),(9,1),(4,5),(5,4),(7,8),(8,7)\}$$

ed il vincolo (56) associato a $S^* = \{4,5\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

5. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,3),(3,6),(6,5),(5,4),(4,1),(7,8),(8,9),(9,7)\}$$

ed il vincolo (56) associato a $S^* = \{7,8,9\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

6. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(2,3),(3,9),(9,6),(6,5),(5,4),(4,1),(7,8),(8,7)\}$$

ed il vincolo (56) associato a $S^* = \{7,8\}$ è violato ed aggiunto a $\tilde{A}x \geq \tilde{b}$

7. le componenti positive di x^* sono date da $x^*_a = 1$ per

$$a \in \{(1,2),(4,5),(7,8)\}$$

e da $x^*_a = 1/2$ per

$$a \in \{(2,3),(2,4),(3,1),(3,6),(5,6),(5,7),(6,4),(6,9),(8,1),(8,9),(9,3),(9,7)\}$$

e nessun vincolo (56) è violato, quindi x^* è anche la soluzione ottima dell'LP completo

Stable Set

Consideriamo ora la soluzione del rilassamento continuo del modello "forte" dello Stable Set (38), (41), (40)

Al passo 1, è naturale definire come LP iniziale il rilassamento continuo del modello "intermedio" con il sottoinsieme delle disuguaglianze (41) corrispondente a (42)

Il problema della separazione da risolvere al passo 3 si riferisce ovviamente alla ricerca di vincoli (41) violati da x^*, e può essere formulato come segue:

Dato $x^* = (x_a^*) \geq 0$, determinare, se esiste, una sottoinsieme $K^* \in \mathcal{K}$ tale che $\sum_{j \in K^*} x_j^* > 1$

Anzichè considerare esplicitamente tutti i possibili sottoinsiemi $K \in \mathcal{K}$ (impraticabile per valori di $|V|$ non troppo piccoli), il problema di separazione può essere risolto risolvendo il problema della Clique di peso massimo con pesi (x_j^*), formulandolo tramite il modello ILP "intermedio" (perfettamente analogo a quello per lo Stable Set) in modo da evitare di avere anche per il problema di separazione un numero esponenziale di vincoli

Indicata con K^* la clique di peso massimo con pesi (x_j^*), se $\sum_{j \in K^*} x_j^* \leq 1$ allora tutti i vincoli (41) sono soddisfatti da x^*, altrimenti una qualsiasi clique massimale K' tale che $K^* \subseteq K'$ (ovviamente $K' = K^*$ se K^* è massimale) definisce il vincolo (41) *più violato* da x^*

Modelli con Molte Variabili e Generazione di Colonne

Verranno considerati ora LP il cui numero di variabili è troppo elevato, quali ad esempio il rilassamento continuo dei modelli (37) per il Bin Packing e (48) per il Vertex Coloring

Si consideri nuovamente il generico problema LP:

$$\begin{array}{rcl} \min c^T x & & \\ Ax & \geq & b \\ x & \geq & 0 \end{array} \tag{74}$$

in cui il numero di variabili è molto grande

Mentre nel caso in cui solo un sottoinsieme di vincoli viene considerato si deve testare l'*ammissibilità* della soluzione dell'LP ridotto, quando si considerano solo un sottoinsieme di variabili si deve testare l'*ottimalità* della soluzione dell'LP ridotto

Il test di ottimalità è possibile grazie al *problema duale*, che come già detto è stato introdotto in questo corso proprio a questo scopo

Il duale di (74) ha la forma:

$$\begin{array}{rcl} \max b^T y & & \\ A^T y & \leq & c \\ y & \geq & 0 \end{array} \tag{75}$$

Precisamente, date le soluzioni ottime primale e duale dell'LP con solo un sottoinsieme di variabili (quindi il cui il duale ha solo un sottoinsieme di vincoli), esse sono entrambe ottime per il problema complessivo se e solo se la soluzione duale è ammissibile per il problema complessivo

Il metodo per risolvere (74) senza dovere fornire tutte le variabili al risolutore, che corrisponde al metodo visto in precedenza per LP con molti vincoli applicato a (75) è il seguente:

1. Inizializza $\tilde{A}$, $\tilde{c}^T$ e $\tilde{x}^T$ con un ("piccolo") sottoinsieme delle colonne di A, c^T, x^T
2. Risolvi l'LP con un sottoinsieme di variabili rispetto a (74):

$$\begin{array}{rcl} \min \tilde{c}^T \tilde{x} & & \\ \tilde{A}\tilde{x} & \geq & b \\ \tilde{x} & \geq & 0 \end{array} \tag{76}$$

   ottenendo la soluzione $\tilde{x}^*$ e la corrispondente soluzione duale y^*
3. Se y^* soddisfa tutti i vincoli in $A^T y \leq c$ allora *STOP* ($\tilde{x}^*$ è la soluzione ottima di (74), ponendo a 0 il valore di tutte le variabili che non compaiono in (76)), altrimenti aggiungi le colonne corrispondenti ad alcuni dei vincoli duali violati a $\tilde{A}$, $\tilde{c}^T$, $\tilde{x}^T$ e vai a 2

Anche in questo caso la scelta dell'LP iniziale non è critica, purchè sia *ammissibile*

La separazione per il problema duale al passo 3 è detta *Generazione di Colonne*, anch'essa da risolvere eventualmente tramite la definizione e soluzione di un opportuno problema, spesso tramite ILP, come illustrato nel seguito per alcuni esempi

Dato una soluzione y^* al passo 3 la quantità $c_j - \sum_{i=1}^{m} a_{ij} y_i^*$ è detta *costo ridotto* (o *relativo*) della variabile x_j – quindi un vincolo duale è violato se e solo se la corrispondente variabile ha costo ridotto negativo

Si osservi che, durante la procedura iterativa di aggiunta di variabili e soluzione dell'LP ridotto, il valore ottimo $\tilde{c}^T \tilde{x}^*$ è sempre *superiore* a quello dell'LP completo, e quindi *non si sa* se sia maggiore o minore rispetto al valore ottimo dell'eventuale ILP di cui l'LP completo è il rilassamento continuo

In base all'osservazione sopra, se la procedura viene interrotta (ad esempio perché si usa un algoritmo euristico per risolvere il problema di generazione di e questo non trova vincoli duali violati) *non* è disponibile un lower bound

Bin Packing

Come primo esempio consideriamo la soluzione del rilassamento continuo del modello (37) per il Bin Packing, il cui duale ha la forma:

$$\begin{aligned} \max & \sum_{i=1}^{m} y_i \\ & \sum_{i \in S} y_i \leq 1, \quad S \in \mathcal{S} \\ & y_i \geq 0, \quad i = 1, \ldots, m \end{aligned} \tag{77}$$

Al passo 1, è naturale inserire nell'LP iniziale le sole variabili corrispondenti ad una soluzione euristica del problema, dopo avere reso eventualmente massimali i sottoinsiemi dati dagli oggetti impaccati in ciascun bin

Il problema di generazione di colonne da risolvere al passo 3 si riferisce quindi alla ricerca di vincoli di (77) violati da y^*, e può essere formulato come segue, mettendo in esplicito la definizione di $\mathcal{S}$:

Dato $y^* = (y_i^*) \geq 0$, determinare, se esiste, un sottoinsieme $S^* \subseteq \{1, \ldots, m\}$ tale che:

- $\sum_{i \in S^*} y_i^* > 1$
- $\sum_{i \in S^*} d_i \leq b$
- S^* è massimale rispetto alla proprietà precedente

Trascurando inizialmente la condizione di massimalità, il problema di separazione può essere risolto introducendo le variabili binarie:

$$z_i := \begin{cases} 1, & \text{se l'oggetto } i \text{ apprtiene a } S^* \\ 0, & \text{altrimenti} \end{cases}$$

e determinando, se esiste, una soluzione al seguente sistema:

$$\sum_{i=1}^{m} y_i^* z_i > 1 \tag{78}$$

$$\sum_{i=1}^{m} d_i z_i \leq b \tag{79}$$

$$z_i \in \{0, 1\}, \quad i = 1, \ldots, m \tag{80}$$

Come prima , la "strana" disuguaglianza con maggiore stretto (78) può essere sostituita dalla funzione obiettivo:

$$\max \sum_{i=1}^{m} y_i^* z_i \tag{81}$$

Il corrispondente ILP (81),(79),(80) è il problema del Knapsack con profitti (y_i^*)

Se la soluzione ottima $\overline{z}$ del Knapsack ha valore ≤ 1, allora *tutti* i vincoli duali (77) sono soddisfatti da y^*, altrimenti dato l'insieme $S^* := \{i \in \{1, \ldots, m\} : \overline{z}_i = 1\}$, un qualsiasi insieme massimale $S' \in \mathcal{S}$ tale che $S^* \subseteq S'$ definisce il vincolo (77) *più violato* da y^*

Esempio 14 Si consideri il caso $n = 5$, $b = 10$, $d = (7, 5, 4, 4, 3)$

Considerando la (pessima, per scopi puramente illustrativi) soluzione iniziale in cui ciascun oggetto è impaccato in un bin differente, e le corrispondenti variabili iniziali per l'LP (dopo aver reso i sottoinsiemi massimali) date da x_S per $S \in \{\{1,5\},\{2,5\},\{3,5\},\{4,5\}\}$, il procedimento sopra esegue le seguenti iterazioni:

1. le componenti positive di $\tilde{x}^*$ e y^* sono date da $\tilde{x}_S^* = 1$ per $S \in \{\{1,5\},\{2,5\},\{3,5\},\{4,5\}\}$ e $y_i^* = 1$ per $i \in \{1,2,3,4\}$; il vincolo (77) associato a $S^* = \{2,3\}$ è violato e la corrispondente variabile $x_{\{2,3\}}$ è aggiunta

2. le componenti positive di $\tilde{x}^*$ e y^* sono date da $\tilde{x}_S^* = 1$ per $S \in \{\{1,5\},\{2,3\},\{4,5\}\}$ e $y_i^* = 1$ per $i \in \{1,2,4\}$; il vincolo (77) associato a $S^* = \{2,4\}$ è violato e la corrispondente variabile $x_{\{2,4\}}$ è aggiunta

3. le componenti positive di $\tilde{x}^*$ e y^* sono date da $\tilde{x}_S^* = 1$ per $S \in \{\{1,5\},\{2,3\},\{4,5\}\}$ e $y_i^* = 1$ per $i \in \{1,3,4\}$; il vincolo (77) associato a $S^* = \{3,4\}$ è violato e la corrispondente variabile $x_{\{3,4\}}$ è aggiunta

4. le componenti positive di $\tilde{x}^*$ e y^* sono date da $\tilde{x}_{\{1,5\}}^* = 1$, $\tilde{x}_S^* = 1/2$ per $S \in \{\{2,3\},\{2,4\},\{3,4\}\}$, $y_1^* = 1$ e $y_i^* = 1/2$ per $i \in \{2,3,4\}$ e nessun vincolo (77) è violato, quindi $\tilde{x}^*$ e y^* sono anche le soluzioni ottime primale e duale dell'LP completo

Vertex Coloring

Consideriamo infine la soluzione del rilassamento continuo del modello (48) per il Vertex Coloring

Il problema di generazione di colonne da risolvere al passo 3 (*perfettamente* analogo al problema di separazione dei vincoli (41) per lo Stable Set) corrisponde alla ricerca di vincoli (49) violati da y^*, e può essere formulato come segue:

Dato $y^* = (y_i^*) \geq 0$, determinare, se esiste, un sottoinsieme $S^* \in \mathcal{S}$ tale che $\sum_{i \in S^*} y_i^* > 1$

Il problema può essere risolto risolvendo il problema dello Stable Set di peso massimo con pesi (y_i^*), formulandolo tramite il modello ILP "intermedio"

Indicato con S^* lo stable set di peso massimo con pesi (y_i^*), se $\sum_{i \in S^*} y_i^* \leq 1$ allora tutti i vincoli (49) sono soddisfatti da y^*, altrimenti un qualsiasi stable set massimale S' tale che $S^* \subseteq S'$ definisce il vincolo (49) *più violato* da y^*

Testi d'Esame

Compito 1

Un'azienda di trasporto deve caricare m camion $\{1, \ldots, m\}$ in modo da servire giornalmente un dato insieme di clienti. Nei camion possono essere caricati n pallet $\{1, \ldots, n\}$, il j-esimo dei quali ha un peso w_j ed un profitto p_j. Per l'i-esimo camion il costo fisso di utilizzazione è pari a c_i e la capacità in peso pari a b_i. Si vogliono caricare alcuni dei pallet sui camion in modo da massimizzare la somma dei profitti dei pallet caricati meno la somma dei costi dei camion utilizzati.

1. Si determini un modello ILP di tipo "descrittivo" in cui vi siano, tra le altre, variabili binarie che dicano se un certo pallet è caricato su un certo camion;
2. Si definisca un algoritmo euristico guidato dal rilassamento continuo del modello definito al punto 1;
3. Si determini un modello ILP con una variabile associata ad ogni possibile caricamento di pallet in ogni camion;
4. Per il modello del punto 3, si definisca un modello ILP per il problema della generazione di colonne per le variabili associate al generico camion i;
5. Si discutano le semplificazioni che si avrebbero ai due punti precedenti qualora i costi e le capacità di tutti i camion fossero uguali.

Compito 2

Un rappresentante deve recarsi da una città ad un altra massimizzando la differenza tra profitti relativi alla visita di altre città e costi relativi alla percorrenza di tratti stradali, con il vincolo che la durata complessiva del viaggio sia non superiore ad un dato tempo T. La rete stradale è descritta da un grafo orientato completo $G = (V, A)$ con n vertici $V = \{1, \ldots, n\}$, dove 1 rappresenta la città di partenza ed n quella di arrivo, ed in cui, per ogni arco $(i, j) \in A$, è specificato il tempo di percorrenza $t_{(i,j)}$ ed il costo di percorrenza $c_{(i,j)}$ e, per ogni vertice $i \in V \setminus \{1, n\}$, è specificato il profitto p_i che si ottiene visitandolo.

1. Si determini un modello ILP per il problema;
2. Si definisca un algoritmo euristico guidato dal rilassamento continuo del modello definito al punto 1;
3. Per il modello del punto 1, si definisca un modello ILP per il problema della separazione di eventuali vincoli il cui numero sia esponenziale in n;
4. Si definisca un algoritmo euristico per il problema di separazione del punto 3 guidato dal rilassamento continuo del modello definito in tale punto;
5. Si consideri la variante del problema in cui si debbano determinare m viaggi, anzichè uno solo, tutti dal vertice 1 al vertice n e di durata non superiore a T. Ogni vertice in $V \setminus \{1, n\}$ può essere visitato al massimo da un viaggio, ottenendo un profitto p_i. Si illustrino un modello ILP descrittivo ed un modello ILP con una variabile per ogni possibile viaggio, unitamente ad un modello ILP per la generazione di colonne per il secondo.

Compito 3

In un reparto di lavorazione occorre decidere l'assegnazione di n lavori $\{1, \ldots, n\}$ ad m macchine $\{1, \ldots, m\}$. In particolare, i vincoli di produzione richiedono che ogni lavoro j sia eseguito da p_j macchine ed ogni macchina i esegua al massimo q_i lavori. Supponendo noto il costo c_{ij} dell'assegnazione del lavoro i alla macchina j, si vuole determinare l'assegnazione ammissibile di costo complessivo minimo.

1. Si determini un modello ILP di tipo "descrittivo" con variabili binarie che dicano se un certo lavoro è eseguito da una certa macchina;

2. Si definisca un algoritmo euristico guidato dal rilassamento continuo del modello definito al punto 1;

3. Si determini un modello ILP con una variabile associata ad ogni possibile assegnamento di lavori ad ogni macchina;

4. Per il modello del punto 3, si definisca un modello ILP per il problema della generazione di colonne per le variabili associate alla generica macchina i, indicando anche un semplice algoritmo per risolvere tale problema.

5. Si discutano i cambiamenti alle risposte ai punti precedenti per la variante in cui ciascun lavoro j debba essere eseguito da una sola macchina (cioè $p_j = 1$), abbia tempo di esecuzione t_{ij} su ciascuna macchina i, ed ogni macchina i, anzichè poter eseguire al massimo q_i lavori, possa lavorare per un tempo non superiore a T_i.

Compito 4

Una società di servizi deve organizzare dei corsi di lingua inglese per i suoi dipendenti, identificati dall'insieme $\{1, \ldots, n\}$. Dato che il livello iniziale di conoscenza della lingua dei dipendenti è molto variabile, si è deciso di suddividerli in classi, con l'obiettivo di minimizzare il numero di classi ed il vincolo che ciascuna classe contenga al più k dipendenti. Inoltre, è stata formata una lista L di coppie di dipendenti imponendo il vincolo ulteriore che, per ogni coppia nella lista, i due dipendenti corrispondenti non possono andare nella stessa classe in quanto i loro livelli di conoscenza sono significativamente diversi.

1. Si determinino uno o più modelli ILP di tipo "descrittivo" con variabili binarie che dicano se un certo impiegato è assegnato ad una certa classe, discutendo brevemente procedure di separazione qualora il numero di vincoli risultasse esponenziale in n;
2. Si definisca un algoritmo euristico guidato dal rilassamento continuo del modello definito al punto 1;
3. Si determini un modello ILP con una variabile associata ad ogni possibile assegnamento di impiegati ad una classe;
4. Per il modello del punto 3, si definisca un modello ILP per il problema della generazione di colonne;
5. Si discutano possibili modelli e/o metodi di soluzione della variante in cui, una volta stabilito il minimo numero di classi necessarie, si vuole minimizzare la differenza tra il numero di impiegati nella classe più numerosa e quello nella classe meno numerosa.

Compito 5

Nel progetto di un database relazionale sono note in precedenza le query (richieste) che gli utenti formuleranno al database, e si deve prendere in considerazione la costruzione di indici per velocizzare il tempo di esecuzione delle query. Si indichino con $\{1, \ldots, n\}$ l'insieme degli indici che possono essere costruiti e con $\{1, \ldots, m\}$ l'insieme delle query. A ciascuna query i si deve rispondere tramite un indice j scelto tra quelli costruiti, in un tempo t_{ij}. Inoltre, l'indice j, se costruito, richiede un tempo di manutenzione u_j ed occupa uno spazio su disco d_j. Si vuole decidere quale insieme di indici costruire e con quale indice rispondere ad ogni query in modo da minimizzare la somma dei tempi di risposta e manutenzione e da garantire che lo spazio occupato su disco dagli indici costruiti non superi b.

1. Si determini un modello ILP di tipo "descrittivo" contenente tra le altre variabili binarie che dicano se ad una certa query si risponde tramite un certo indice;
2. Si definisca un algoritmo euristico guidato dal rilassamento continuo di uno dei modelli definiti al punto 1;
3. Si determini un modello ILP con una variabile associata ad ogni possibile assegnamento di query ad ogni indice;
4. Per il modello del punto 3, si definisca un modello ILP per il problema della generazione di colonne per le variabili associate al generico indice j;
5. Si discutano i cambiamenti ai modelli precedenti nel caso in cui ogni query i abbia anche una frequenza f_i, e la somma delle frequenze delle query associate ad un indice j non possa essere superiore a g_j.

Compito 6

Un'azienda di trasporti deve organizzare le consegne ad un insieme $\{1, \ldots, n\}$ di clienti localizzati in una rete stradale, rappresentata da un grafo orientato completo orientato $G = (V, A)$, con $V = \{1, \ldots, n\}$. Per motivi organizzativi, l'azienda decide di suddividere i clienti determinando un insieme di m circuiti disgiunti, ciascuno contenente esattamente k clienti (si supponga $n = m \cdot k$), in modo che ogni cliente sia contenuto in esattamente un circuito, e *senza* un deposito comune, in quanto si vogliono utilizzare m veicoli ciascuno con base presso uno dei clienti serviti. Indicando con c_{ij} il costo dell'arco (i, j), l'obiettivo è la minimizzazione del costo complessivo dei circuiti.

1. Si determini un modello ILP per il problema tramite variabili con uno degli indici riferito al circuito di appartenenza;
2. Si definisca un algoritmo euristico guidato dal rilassamento continuo del modello definito al punto 1;
3. Per il modello del punto 1, si definisca un modello ILP per il problema della separazione di eventuali vincoli il cui numero sia esponenziale in n e relativi al generico circuito h;
4. Si determini un modello ILP con una variabile associata ad ogni possibile circuito;
5. Per il modello del punto 4, si definisca un modello ILP per il problema della generazione di colonne.

Applicazioni

Nella maggior parte dei casi, i problemi illustrati precedentemente non si ritrovano nella realtà esattamente nella forma descritta (pur con eccezioni), ma compaiono come rilassamenti/semplificazioni di problemi più complessi

In questa sezione si vedranno esempi di problemi reali per cui sono definibili modelli ILP, ispirati in buona parte a quelli già visti, utili per la loro soluzione

Per molti altri problemi reali ancor più complessi, anzichè insistere nel definire modelli ILP completi che risulterebbero difficili o impossibili da risolvere perché "deboli", si preferisce definire modelli ILP per opportuni rilassamenti (eliminando o semplificando parte dei vincoli) ed utilizzare le informazioni fornite dalla soluzione di questi ILP e/o dai loro rilassamenti continui per guidare algoritmi euristici

Turnazione del Personale

I problemi di turnazione del personale sono probabilmente l'applicazione al momento più rilevante dell'Ottimizzazione (e quindi dei modelli ILP)

Da un lato, in svariati casi, i vincoli imposti sui turni sono molto complessi, con l'effetto che una costruzione dei turni "non accurata" porta facilmente a soluzioni "cattive" rispetto a quelle ottime

Dall'altro, i costi in gioco sono molto elevati, e quindi la determinazione di soluzioni ottime (o quasi) consente alle aziende risparmi considerevoli

Generalmente, le soluzioni ottenute "manualmente" senza l'ausilio dell'Ottimizzazione sono di "buona" qualità, ma richiedono tempi e costi elevati per la loro definizione (lavoro di molti turnisti esperti per svariate settimane)

Tramite l'uso di modelli ILP, si ottengono soluzioni di qualità generalmente migliore di quelle ottenute "manualmente" in tempi molto ridotti – questo consente di valutare velocemente l'effetto di cambiamenti (aggiunta di nuovi servizi, variazione dei vincoli lavorativi) sui turni, cosa assolutamente impensabile agendo "manualmente"

Definizione e classificazione

La casistica dei problemi di turnazione del personale è molto vasta, cosa che rende impossibile una classificazione precisa

Tutti questi problemi richiedono di determinare, per un periodo dato (ad esempio 6 mesi) ed un insieme di *servizi* che devono essere svolti in tale periodo, il numero di impiegati da utilizzare ed il *turno* di ciascun impiegato, cioè esattamente cosa l'impiegato farà nel periodo dato

A grandi linee, i problemi possono essere suddivisi in:

1. Problemi in cui l'orario di lavoro giornaliero è scelto all'interno di un numero "piccolo" di possibilità (ad esempio, uno tra 6-14, 14-22, 22-6)

2. Problemi in cui l'orario di lavoro giornaliero può essere definito in moltissimi modi, data la struttura complessa dei servizi da svolgere (come nel caso dei turni del personale di compagnie di trasporto), a loro volta suddivisi in:

   (a) Problemi in cui l'insieme di servizi da svolgere complessivamente di notte e nel fine settimana è significativo (come nel caso del trasporto ferroviario)

   (b) Problemi in cui l'insieme di servizi da svolgere complessivamente di notte e nel fine settimana è trascurabile o nullo (come nel caso del trasporto urbano)

L'appartenenza ad un tipo tra quelli sopra influenza fortemente la complessità dei problemi incontrati

Per *carico lavorativo giornaliero* si intende la definizione degli orari di lavoro (unitamente ai servizi da svolgere) che dovranno effettuare gli impiegati che lavoreranno in quel giorno, *senza* specificare di quali impiegati faranno quale orario o si riposeranno

- Per i problemi del tipo 1, la definizione del carico lavorativo giornaliero è data o facilmente definibile, mentre la parte complessa è la definizione dei turni di lavoro a partire da questo carico

- Per i problemi del tipo 2b la parte complessa è la definizione del carico lavorativo giornaliero, mentre, una volta specificato questo, la definizione dei turni è molto semplice

- Per i problemi del tipo 2a entrambe le parti di definizione del carico lavorativo giornaliero e di definizione dei turni di lavoro a partire da questo carico sono complesse

Turnazione del Personale in Aziende di Trasporto

Nelle aziende di trasporto i *servizi* da svolgere corrispondono a *corse* di un veicolo da un punto ad un altro, che richiedono un *equipaggio di guida* (ad esempio due macchinisti) ed un *equipaggio di bordo* (ad esempio due controllori)

Per semplicità, faremo riferimento al caso dell'equipaggio di guida, indicato come *guidatore*

È prassi comune suddividere il problema di turnazione in due *fasi*:

- *Fase I* in cui, a partire dalle corse si definisce il carico lavorativo giornaliero

- *Fase II* in cui, a partire dal carico lavorativo giornaliero si definiscono i turni

I singoli componenti del carico lavorativo giornaliero, ovverosia una sequenza di servizi che devono essere effettuati da un guidatore in un giorno, sono detti *pairing*

Fase I

La definizione dei pairing a partire dalle corse, sempre soggetta a vincoli complicati, è generalmente affrontata tramite modelli di tipo Set Partitioning/Set Covering

Si assuma per semplicità che l'insieme delle corse da effettuare non cambi per ogni giorno in cui si deve lavorare (ad esempio, tutti i giorni ad eccezione della Domenica)

Si assuma di aver specificato:

- $\{1, \ldots, m\}$ l'insieme delle corse da coprire ogni giorno
- $\mathcal{S}$ la collezione di tutti i possibili pairing ottenibili dalle corse date
- per ciascun pairing $S \in \mathcal{S}$, il *costo* c_S ed il corrispondente insieme delle corse effettuate

Un modello di tipo Set Partitioning si ottiene definendo le variabili:

$$x_S := \begin{cases} 1, & \text{se il pairing } S \text{ è selezionato in soluzione} \\ 0, & \text{altrimenti} \end{cases}$$

ed ha la forma:

$$\begin{aligned} \min & \sum_{S \in \mathcal{S}} c_S x_S \\ & \sum_{S \in \mathcal{S}: S \text{ effettua } i} x_S = 1, \quad i = 1, \ldots, m \\ & x_S \in \{0, 1\}, \quad S \in \mathcal{S} \end{aligned} \tag{82}$$

In molte applicazioni, si utilizza il corrispondente modello di tipo Set Covering, che ha una soluzione migliore (o uguale) in quanto meno vincolato, consentendo di assegnare *più guidatori* ad una stessa corsa, facendo guidare uno solo di essi e trasportando gli altri come passeggeri

Generalmente solo in *applicazioni di trasporto aereo*, in cui i posti sono limitati e preziosi, si utilizza il modello di tipo Set Partitioning

Per quanto riguarda la gestione di $\mathcal{S}$:

- In alcuni casi è possibile *introdurre esplicitamente* tutti i pairing possibili
- In altri casi viene utilizzata una procedura di generazione di colonne, che spesso è di tipo euristico essendo i vincoli imposti sui pairing molto complessi

- In altri casi ancora, volendo evitare la generazione di colonne ma non potendo introdurre tutti i pairing, viene creata *inizialmente* una sottocollezione di pairing "buoni" e si risolve il problema con solo questi pairing

Fase II

Come già detto, la definizione dei turni a partire dai pairing è talvolta molto semplice e talvolta complessa – in questo secondo caso i vincoli imposti sui turni sono piuttosto complicati ed i metodi di soluzione variano molto da caso a caso, per cui in questo corso non verranno presentati esempi

Turnazione del Personale in un Call Center

Come caso di studio concreto descritto in dettaglio, viene considerato il problema della turnazione del personale in un call center

Il problema richiede di determinare i turni per un periodo di n giorni, con il seguente carico di lavoro:

- Sono dati n_q *orari*, ciascuno corrispondente ad un dato orario di lavoro all'interno della giornata
- Per ogni orario q e giorno j, è richiesto che *almeno* e_{qj} impiegati svolgano l'orario q nel giorno j

L'obiettivo è la copertura degli orari con il *minimo* numero di turni, ciascuno da assegnare ad un diverso impiegato (per cui i termini turno ed impiegato possono essere usati come sinonimi ai fini del problema)

La struttura di un turno è soggetta ai seguenti vincoli:

- I turno si ripete *ciclicamente* ogni n giorni
- Nel turno si alternano *blocchi lavorativi* e *riposi*
- Il turno contiene b blocchi lavorativi, ciascuno corrispondente a k giorni di lavoro consecutivi *con lo stesso orario*
- Ci sono n_r *tipi di riposi* differenti
- Per ciascun tipo di riposo r, il turno contiene a_r riposi di tipo r, ciascuno corrispondente a d_r giorni di riposo consecutivi

Si osservi che i dati sopra sono coerenti solo se $b = \sum_{r=1}^{n_r} a_r$ e $bk + \sum_{r=1}^{n_r} a_r d_r = n$

I blocchi lavorativi possono essere spezzati in due parti, una comprendente i primi giorni del periodo ed una gli ultimi – in questo caso *non è necessario* che l'orario di una parte coincida con quello dell'altra

Esempio 15 Si consideri il caso $n = 14$; $n_q = 4$, $e_{qj} = 1$ per $q = 1, 2, 3$ e $j = 1, \ldots, 14$, $e_{4j} = 1$ per $j \neq 7, 14$ ed $e_{4j} = 0$ per $j = 7, 14$; $b = 3$, $k = 3$, $n_r = 2$, $a_1 = 2$, $d_1 = 1$, $a_2 = 1$, $d_2 = 3$ (in cui 3 dei 4 orari sono da svolgere tutti i giorni mentre il quarto non deve essere svolto la Domenica)

Verificare che 6 turni sono sufficienti a patto di assegnare orari diversi a blocchi spezzati tra inizio e fine periodo

Indicando con $f_j := \sum_{q=1}^{n_q} e_{qj}$ il minimo numero di turni che devono lavorare il giorno j, il metodo risolutivo proposto per il problema lo suddivide in due parti:

- Parte I, in cui vengono stabiliti i *pattern lavorativi*, corrispondenti a turni in cui *l'orario dei blocchi lavorativi non è assegnato*, garantendo che almeno f_j turni lavorino per ogni giorno j e minimizzando il numero di turni
- Parte II, in cui vengono assegnati gli orari ai blocchi lavorativi dei pattern determinati nella Parte I, garantendo che almeno e_{qj} turni lavorino con l'orario q per ogni giorno j

La Parte I corrisponde ad un problema di ottimizzazione, mentre nella Parte II ci si limita a cercare una soluzione ammissibile

Si osservi che, se nella Parte II si trova una soluzione ammissibile, questa è necessariamente ottima per il problema complessivo (cioè si utilizzano il minimo numero di turni)

Dato che è facile nei casi reali determinare una soluzione nella Parte II, non verrà illustrato come procedere nel caso in cui questa non si trovi

Parte I

Per questa parte vedremo un modello "descrittivo" con variabili che specificano la struttura dei pattern, ed un modello di tipo Set Covering con una variabile per ogni possibile pattern

Nel modello "descrittivo", si assuma di avere m impiegati (pattern) disponibili, dove m è una stima sul numero di impiegati – qualora questo valore risulti essere troppo piccolo (e quindi il modello senza soluzione) occorrerà aumentarlo; d'altra parte la dimensione del modello è proporzionale ad m che quindi non deve essere eccessivamente grande

Oltre alle variabili naturali:

$$y_i := \begin{cases} 1, & \text{se il pattern } i \text{ viene utilizzato} \\ 0, & \text{altrimenti} \end{cases}$$

si sarebbe tentati di utilizzare le variabili:

$$x_{ij} := \begin{cases} 1, & \text{se il pattern } i \text{ lavora nel giorno } j \\ 0, & \text{altrimenti} \end{cases}$$

Sfortunatamente, con le variabili x definite come sopra è estremamente complesso imporre il vincolo che i blocchi lavorativi ed i riposi sono formati da un certo numero di giorni consecutivi

Le variabili x che consentono di ottenere un modello relativamente semplice sono invece:

$$x_{ij} := \begin{cases} 1, & \text{se il pattern } i \textit{ inizia un blocco lavorativo} \text{ nel giorno } j \\ 0, & \text{altrimenti} \end{cases}$$

unitamente alla loro controparte per i riposi:

$$w_{ij}^r := \begin{cases} 1, & \text{se il pattern } i \textit{ inizia un riposo di tipo } r \text{ nel giorno } j \\ 0, & \text{altrimenti} \end{cases}$$

Con queste ultime variabili x e w, assieme alle variabili y, notando che un pattern i lavora nel giorno j se e solo se $\sum_{h=j-k+1}^{j} x_{ih} = 1$, il modello ha la forma:

$$\min \sum_{i=1}^{m} y_i \tag{83}$$

$$\sum_{i=1}^{m} \sum_{h=j-k+1}^{j} x_{ih} \geq f_j, \quad j = 1, \ldots, n \tag{84}$$

$$\sum_{j=1}^{n} x_{ij} = by_i, \quad i = 1, \ldots, m \tag{85}$$

$$\sum_{j=1}^{n} w_{ij}^r = a_r y_i, \quad i = 1, \ldots, m,\ r = 1, \ldots, n_r \tag{86}$$

$$\sum_{h=j-k+1}^{j} x_{ih} + \sum_{r=1}^{n_r} \sum_{h=j-d_r+1}^{j} w_{ih}^r = y_i, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{87}$$

$$x_{ij} = \sum_{r=1}^{n_r} w_{i(j-d_r)}^r, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{88}$$

$$x_{i(j-k)} = \sum_{r=1}^{n_r} w_{ij}^r, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{89}$$

$$y_i, x_{ij}, w_{ij}^r \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n,\ r = 1, \ldots, n_r \tag{90}$$

I vincoli (85)-(89) garantiscono che il pattern sia ammissibile – in particolare (87) impongono che ogni giorno il pattern lavori o riposi, (88) che prima di ogni blocco vi sia un riposo, ed i vincoli (89) che dopo ogni blocco vi sia un riposo

Il modello visto, che è fortemente simmetrico, pur consentendo di risolvere casi piccoli, è nettamente peggiore del modello che segue (anche se i lower bound dei due rilassamenti continui generalmente coincidono), che è necessario utilizzare per risolvere i casi reali

Nel secondo modello, indicando con $\mathcal{P}$ la collezione di tutti i pattern ammissibili e notando che potrebbe essere conveniente utilizzare uno stesso pattern più volte nella soluzione (cioè avere impiegati che lavorano secondo lo stesso pattern), si utilizzano le variabili:

$$x_P := \text{numero di impiegati che lavorano secondo il pattern } P$$

Indicando con $\mathcal{P}_j \subseteq \mathcal{P}$ la sottocollezione di pattern che lavorano il giorno j, il modello, analogo al Set Covering ma con variabili intere e termini noti interi generici, ha la forma:

$$\min \sum_{P \in \mathcal{P}} x_P \tag{91}$$

$$\sum_{P \in \mathcal{P}_j} x_P \geq f_j, \quad j = 1, \ldots, n \tag{92}$$

$$x_P \geq 0, \text{intero}, \quad P \in \mathcal{P} \tag{93}$$

In alcuni casi pratici, il numero di possibili pattern $|\mathcal{P}|$ è sufficientemente contenuto da consentire la costruzione esplicita di tutti questi pattern e la soluzione diretta del modello sopra

In altri casi, occorre risolvere il rilassamento continuo del problema ricorrendo alla generazione di colonne, che richiede di considerarne il duale, nel quale si indica con J_P l'insieme dei giorni in cui un pattern $P \in \mathcal{P}$ lavora:

$$\max \sum_{j=1}^{n} f_j y_j \tag{94}$$

$$\sum_{j \in J_P} y_j \leq 1, \quad P \in \mathcal{P} \tag{95}$$

$$y_j \geq 0, \quad j = 1, \ldots, n \tag{96}$$

Il problema di generazione di colonne richiede, dato $y^* = (y_j^*)$, di determinare, se esiste, un pattern $P \in \mathcal{P}$ tale che:

$$\sum_{j \in J_P} y_j^* > 1$$

Un tale pattern P esiste se e solo se il valore ottimo dell'ILP che segue è > 1 (ed in tal caso è indicato dalla soluzione dell'ILP)

Dovendo definire un pattern, la struttura del modello è simile a quella del modello "descrittivo" visto sopra, con la differenza che scompare l'indice relativo al pattern (si cerca un solo pattern) e la funzione obiettivo richiede di massimizzare la somma dei "profitti" y_j^* dei giorni in cui il pattern lavora

Si considerino le variabili:

$$z_j := \begin{cases} 1, & \text{se il pattern } P \textit{ inizia un blocco lavorativo} \text{ nel giorno } j \\ 0, & \text{altrimenti} \end{cases}$$

unitamente alla loro controparte per i riposi:

$$u_j^r := \begin{cases} 1, & \text{se il pattern } P \textit{ inizia un riposo di tipo } r \text{ nel giorno } j \\ 0, & \text{altrimenti} \end{cases}$$

Il modello ILP per la generazione di colonne ha la forma:

$$\max \sum_{j=1}^{n} y_j^* \left(\sum_{h=j-k+1}^{j} z_h \right) \tag{97}$$

$$\sum_{j=1}^{n} z_j = b, \tag{98}$$

$$\sum_{j=1}^{n} u_j^r = a_r, \quad r = 1, \ldots, n_r \tag{99}$$

$$\sum_{h=j-k+1}^{j} z_h + \sum_{r=1}^{n_r} \sum_{h=j-d_r+1}^{j} u_h^r = 1, \quad j = 1, \ldots, n \tag{100}$$

$$z_j = \sum_{r=1}^{n_r} u_{j-d_r}^r, \quad j = 1, \ldots, n \tag{101}$$

$$z_{j-k} = \sum_{r=1}^{n_r} u_j^r, \quad j = 1, \ldots, n \tag{102}$$

$$z_j, u_j^r \in \{0,1\}, \quad j = 1, \ldots, n, \ r = 1, \ldots, n_r \tag{103}$$

Tramite il secondo modello ILP per la Parte I, generando le colonne tramite il modello ILP sopra, si risolvono facilmente casi con centinaia di giorni e di impiegati necessari

Parte II

Come già detto, obiettivo della Parte II è determinare una soluzione ammissibile e, qualora vi si riesca, la soluzione finale determinata per il problema completo è certamente ottima

Qualora in questa parte non si trovi una soluzione occorre ridefinire i pattern nella Parte I *in qualche modo* – dato però che questo non accade in pratica (come già detto) non verrà discusso questo caso

Un semplice modello ILP per la Parte II si ottiene osservando che, in questa fase, ciò che importa conoscere è l'insieme complessivo dei blocchi lavorativi per i pattern determinati nella Parte I (senza bisogno di sapere a quale pattern appartenga ciascun blocco lavorativo)

Si indichi con n_b il numero di blocchi lavorativi complessivi, considerando come *due* blocchi lavorativi differenti le due parti di un blocco lavorativo che contiene i giorni 1 e n (in quanto alle due parti possono essere assegnati orari diversi), e si introducano le variabili:

$$x_{bq} := \begin{cases} 1, & \text{se il blocco lavorativo } b \text{ lavora secondo l'orario } q \\ 0, & \text{altrimenti} \end{cases}$$

Indicando con B_j l'insieme dei blocchi lavorativi che lavorano il giorno j, un semplice modello ILP (senza funzione obiettivo) che in pratica consente facilmente di trovare una soluzione è:

$$\sum_{q=1}^{n_q} x_{bq} = 1, \quad b = 1, \ldots, n_b \tag{104}$$

$$\sum_{b \in B_j} x_{bq} \geq e_{qj}, \quad q = 1, \ldots, n_q, \; j = 1, \ldots, n \tag{105}$$

$$x_{bq} \in \{0, 1\}, \quad b = 1, \ldots, n_b, \; q = 1, \ldots, n_q \tag{106}$$

Trasporto Merci

I problemi di trasporto merci sono sempre stati di grande importanza pratica, e tendono ad esserlo ancora di più con la diffusione delle vendite on-line

In questa sede ci limiteremo al caso in cui i trasporto avviene su strada come è (purtroppo) nella maggior parte dei casi, e come è inevitabile che sia su "piccola scala" (da urbana a regionale)

In un generico problema è dato un certo insieme di *clienti* che devono essere serviti da parte di *veicoli* che hanno base presso uno o più *depositi* e si muovono in una *rete stradale*

Si vuole determinare il viaggio che deve effettuare ciascun veicolo in modo da servire i clienti a costo minimo

La rete stradale è rappresentata da un grafo orientato in cui i vertici sono i cosiddetti *punti notevoli* della rete, vale a dire i clienti, i depositi e gli *incroci*, e gli archi i collegamenti tra questi punti, ciascuno con un associato costo di percorrenza, generalmente corrispondente al *tempo* di percorrenza

Ciascun cliente è caratterizzato da:

- il vertice corrispondente nella rete stradale
- la quantità ed il tipo di merce che deve essere consegnata al cliente/raccolta presso il cliente, con eventuali tempi di scarico/carico
- un'eventuale *finestra temporale* all'interno della quale deve essere servito
- eventuali restrizioni sui veicoli che lo possono servire
- un'eventuale *penalità* qualora non sia servito

Ciascun veicolo è caratterizzato da:

- il deposito di appartenenza ed il corrispondente vertice nella rete stradale
- la *capacità* di carico, intesa come quantità e tipo di merce trasportabile dal veicolo
- eventuali vincoli sull'orario di lavoro dell'autista associato

Gli obiettivi più frequenti sono generalmente dati da uno tra:

- la minimizzazione dei costi di viaggio con il vincolo di servire tutti i clienti
- la minimizzazione delle penalità dei clienti non serviti con il vincolo che il costo di viaggio complessivo non superi un dato budget

Per rappresentare e risolvere questi problemi, il grafo corrispondente alla rete stradale viene ridotto ad un grafo *completo* $G = (V, A)$ in cui i vertici sono i soli clienti e depositi (eliminando tutti gli incroci) ed il costo di ciascun arco (i, j) è dato dal minimo costo per andare da i a j nella rete stradale (cammino minimo)

(Nel grafo ridotto i costi soddisfano la disuguaglianza triangolare (50))

La specificazione esatta delle caratteristiche illustrate sopra porta ad un'ampia varietà di problemi di interesse pratico

In questo ambito ci limiteremo ad un problema *base*, chiamato (Capacitated) Vehicle Routing, in cui:

- vi è un unico tipo di merce che deve essere solo consegnata (o solo raccolta)
- vi è un solo deposito, e corrispondentemente $V = \{0, 1, \ldots, n\}$, con 0 che rappresenta il deposito ed $1, \ldots, n$ che rappresentano i clienti
- la domanda di ciascun cliente j è rappresentata da uno scalare d_j

- sono dati k veicoli uguali, ciascuno con capacità b

Si vuole determinare, per ogni veicolo, un *circuito* che parta dal e ritorni al deposito in modo che:

- ciascun cliente sia servito da un veicolo (ovverosia, tenendo presente la disuguaglianza triangolare (50), ciascun vertice eccetto 0 sia visitato *esattamente* da un circuito)
- la somma delle domande dei clienti serviti da ciascun veicolo (ovverosia dei vertici visitati da ciascun circuito) non ecceda b
- la somma dei costi dei circuiti sia minimizzata

Si osservi che il problema ammette soluzione se e solo se il problema di Bin Packing con n oggetti con pesi (d_j) ha una soluzione che utilizza k bin

Per il Vehicle Routing, verranno illustrati diversi modelli ILP, ciascuno dei quali viene utilizzato in pratica

Esempio 16 Si consideri il caso $n = 6$, $b = 10$, $d = (7, 5, 4, 4, 2)$ e lo si utilizzi come esempio di riferimento per la comprensione dei modelli visti

Modello 1

Il primo modello, analogo a quello visto per l'ATSP, utilizza variabili che specificano se un dato arco appartiene ad un qualche circuito in soluzione senza specificare di quale circuito si tratta:

$$x_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene ad uno dei } k \text{ circuiti} \\ 0, & \text{altrimenti} \end{cases}$$

La parte "facile" del modello ha la forma:

$$\min \sum_{a \in A} c_a x_a \tag{107}$$

$$\sum_{a \in \delta^-(i)} x_a = 1, \quad i \in V \setminus \{0\} \tag{108}$$

$$\sum_{a \in \delta^+(i)} x_a = 1, \quad i \in V \setminus \{0\} \tag{109}$$

$$\sum_{a \in \delta^-(0)} x_a = k, \tag{110}$$

$$\sum_{a \in \delta^+(0)} x_a = k, \tag{111}$$

$$x_a \in \{0,1\}, \quad a \in A \tag{112}$$

A questi si devono aggiungere vincoli per:

- eliminare circuiti che non passino per il deposito
- eliminare circuiti che non rispettino il vincolo di capacità

I due obiettivi sono raggiunti aggiungendo la seguente variante (complicata) dei vincoli (57)

Per un dato sottoinsieme $S \subseteq \{1, \ldots, n\}$ di clienti, si indichi con $\sigma(S)$ il valore della soluzione ottima del problema di bin packing con $|S|$ oggetti con pesi dati dalle domande dei clienti in S e con capacità dei bin data dalla capacità b dei veicoli

I vincoli da aggiungere per completare il modello sono:

$$\sum_{a \in \delta^+(S)} x_a \geq \sigma(S), \quad S \subseteq V \setminus \{0\} \tag{113}$$

La separazione di questi vincoli è facile se la soluzione $x^* = (x_a^*)$ è intera (basta verificare se la soluzione contiene un circuito che non passa per il deposito oppure un circuito che non rispetta il vincolo di capacità), mentre è estremamente complessa in caso contrario, ed in pratica ci si limita a considerare algoritmi euristici

Una variante meno "forte" dei vincoli (113) che fornisce comunque un modello ILP valido e che può essere separata tramite un modello ILP analogo a quello illustrato per la separazione dei vincoli (56) per l'ATSP è data da:

$$\sum_{a \in \delta^+(S)} x_a \geq \frac{\sum_{j \in S} d_j}{b}, \quad S \subseteq V \setminus \{0\} \tag{114}$$

La separazione dei vincoli (114) richiede, dato $x^* = (x_a^*)$, di determinare, se esiste, un sottoinsieme $S \subseteq V \setminus \{0\}$ tale che:

$$\sum_{j \in S} d_j - b \sum_{a \in \delta^+(S)} x_a^* > 0$$

Definendo le variabili binarie:

$$y_i := \begin{cases} 1, & \text{se il vertice } i \text{ appartiene a } S \\ 0, & \text{altrimenti} \end{cases}$$

e

$$z_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene a } \delta^+(S) \\ 0, & \text{altrimenti} \end{cases}$$

esiste un vincolo violato corrispondente all'insieme S se e solo se il valore ottimo del seguente ILP è > 0:

$$\max \sum_{i \in V} d_i y_i - b \sum_{a \in A} x_a^* z_a \tag{115}$$

$$\begin{aligned} y_0 &= 0 && (116) \\ z_{(i,j)} &\leq y_i, \quad (i,j) \in A && (117) \\ z_{(i,j)} &\leq 1 - y_j, \quad (i,j) \in A && (118) \\ z_{(i,j)} &\geq y_i - y_j, \quad (i,j) \in A && (119) \\ y_i, z_a &\in \{0,1\}, \quad i \in V,\ a \in A && (120) \end{aligned}$$

Modello 2

Il secondo modello, che a differenza del primo si adatta facilmente al caso in cui i veicoli abbiano caratteristiche differenti, utilizza variabili che specificano se un dato arco appartiene ad un dato circuito:

$$x_a^h := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene al circuito } h \\ 0, & \text{altrimenti} \end{cases}$$

Inoltre, vengono utilizzate anche variabili che dicono se un dato vertice in $V \setminus \{0\}$ è visitato da un dato circuito:

$$y_i^h := \begin{cases} 1, & \text{se il vertice } i \text{ è visitato dal circuito } h \\ 0, & \text{altrimenti} \end{cases}$$

Il modello ha la forma:

$$\min \sum_{h=1}^{k} \sum_{a \in A} c_a x_a^h \tag{121}$$

$$\sum_{a \in \delta^-(i)} x_a^h = y_i^h, \quad h = 1, \ldots, k, \ i \in V \setminus \{0\} \tag{122}$$

$$\sum_{a \in \delta^+(i)} x_a^h = y_i^h, \quad h = 1, \ldots, k, \ i \in V \setminus \{0\} \tag{123}$$

$$\sum_{h=1}^{k} y_i^h = 1, \quad i \in V \setminus \{0\} \tag{124}$$

$$\sum_{a \in \delta^-(0)} x_a^h = 1, \quad h = 1, \ldots, k \tag{125}$$

$$\sum_{a \in \delta^+(0)} x_a^h = 1, \quad h = 1, \ldots, k \tag{126}$$

$$\sum_{i \in V \setminus \{0\}} d_i y_i^h \leq b, \quad h = 1, \ldots, k \tag{127}$$

$$\sum_{a \in \delta^+(S)} x_a^h \geq y_i^h, \quad h = 1, \ldots, k, \ S \subseteq V \setminus \{0\}, \ i \in S \tag{128}$$

$$x_a^h, y_i^h \in \{0, 1\}, \quad a \in A, \ i \in V \setminus \{0\} \tag{129}$$

(Come mostrato dal modello, ciascuna variabile y è definita da una somma di variabili x, e quindi sarebbe facilmente eliminabile – le variabili y sono introdotte per maggiore leggibilità)

Mentre i vincoli (127) chiaramente impongono il vincolo di capacità, i vincoli (128), che sono una variante dei vincoli (57) per l'ATSP, proibiscono sottocircuiti, imponendo che, se un vertice di un sottoinsieme S dei clienti è visitato dal circuito h, allora esiste almeno un arco uscente da S verso i rimanenti vertici (che includono il deposito)

La separazione dei vincoli (128) è simile alla separazione dei vincoli (57) per l'ATSP, con una differenza concettuale sostanziale

Il problema di separazione richiede, dati $\overline{x} = (\overline{x}_a^h)$ e $\overline{y} = (\overline{y}_i^h)$, di determinare, se esistono, un circuito h, un sottoinsieme $S \subseteq V \setminus \{0\}$ ed un vertice $i \in S$ tali che:

$$\sum_{a \in \delta^+(S)} \overline{x}_a^h < \overline{y}_i^h$$

Per risolvere questo problema si considerano, uno dopo l'altro, tutti gli indici $h = 1, \ldots, k$, cercando un vincolo violato associato al circuito h

Inoltre, fissato h, si considerano, uno dopo l'altro tutti i vertici $i \in V \setminus \{0\}$, cercando un vincolo violato associato al circuito h e al vertice i

Una volta fissati h ed i, si determina S risolvendo il seguente ILP (quindi complessivamente per la separazione si risolvono kn ILP)

Definendo le variabili binarie:

$$w_j := \begin{cases} 1, & \text{se il vertice } j \text{ appartiene a } S \\ 0, & \text{altrimenti} \end{cases}$$

e

$$z_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene a } \delta^+(S) \\ 0, & \text{altrimenti} \end{cases}$$

esiste un vincolo violato corrispondente al circuito h, al vertice i e all'insieme S se e solo se il valore ottimo del seguente ILP è $< \overline{y}_i^h$:

$$\min \sum_{a \in A} x_a^* z_a \tag{130}$$

$$\begin{aligned} y_0 &= 0 && (131)\\ y_i &= 1 && (132)\\ z_{(i,j)} &\leq y_i, \quad (i,j) \in A && (133)\\ z_{(i,j)} &\leq 1 - y_j, \quad (i,j) \in A && (134)\\ z_{(i,j)} &\geq y_i - y_j, \quad (i,j) \in A && (135)\\ y_i, z_a &\in \{0,1\}, \quad i \in V,\ a \in A && (136) \end{aligned}$$

Come esempio dell'uso del modello visto, consideriamo il caso in cui, oltre ai vincoli considerati finora, siano anche imposti vincoli di *finestre temporali* che impongono che ciascun cliente i sia visitato all'interno dell'intervallo di tempo $[a_i, b_i]$, assumendo che:

- tutti i veicoli partano dal deposito all'istante 0
- i tempi di viaggio da un vertice all'altro siano pari ai costi (c_a)
- il tempo di scarico presso ciascun cliente sia nullo
- nessun veicolo possa rallentare o fermarsi in modo che il tempo di percorrenza del circuito corrispondente sia superiore al suo costo

In questo caso, i vincoli di finestre temporali possono essere imposti aggiungendo le variabili:

$$s_i^h := \begin{cases} \text{istante di visita del vertice } i \text{ nel circuito } h \text{ se } y_i^h = 1 \\ 0 \text{ altrimenti} \end{cases}$$

ed i vincoli:

$$\begin{aligned} s_i^h &\geq a_i y_i^h, \quad h = 1, \ldots, k,\ i \in V \setminus \{0\} && (137) \\ s_i^h &\leq b_i y_i^h, \quad h = 1, \ldots, k,\ i \in V \setminus \{0\} && (138) \\ s_0^h &= 0, \quad h = 1, \ldots, k && (139) \\ x_{(i,j)}^h = 1 &\Rightarrow s_j^h = s_i^h + c_{(i,j)}, \quad h = 1, \ldots, k,\ i \in V,\ j \in V \setminus \{0\} && (140) \end{aligned}$$

Per imporre i vincoli logici (140) tramite disuguaglianze lineari, è necessario utilizzare un valore costante grande M (detto *"big" M*) per disattivare il vincolo qualora $x_{(i,j)}^h = 0$, scrivendo:

$$\begin{aligned} s_j^h &\geq s_i^h + c_{(i,j)} - M(1 - x_{(i,j)}^h), \quad h = 1, \ldots, k,\ i \in V,\ j \in V \setminus \{0\} && (141) \\ s_j^h &\leq s_i^h + c_{(i,j)} + M(1 - x_{(i,j)}^h), \quad h = 1, \ldots, k,\ i \in V,\ j \in V \setminus \{0\} && (142) \end{aligned}$$

È facile costruire esempi che mostrano la "debolezza" del rilassamento continuo in presenza del "big" M, nel senso che basta che la variabile $x_{(i,j)}^h$ si discosti poco da un valore intero per disattivare i due vincoli sopra

In pratica, bisognerebbe evitare per quanto possibile modelli ILP con il "big" M, ma d'altra parte ci sono casi come quello mostrato (o come molti dei problemi di Scheduling illustrati nel seguito) in cui non se ne riesce a fare a meno

Se il "big" M non è evitabile, è comunque opportuno definire un valore di M *differente* per ogni vincolo, pari al minimo valore che rende il vincolo valido, ed essere coscienti del fatto che si sta lavorando con un modello intrinsecamente "debole"

Modello 3

L'ultimo modello illustrato per il Vehicle Routing è analogo ai modelli di tipo Set Packing visti per il Bin Packing ed il Vertex Coloring, e viene prevalentemente utilizzato quando ci sono vincoli di natura complessa imposti sui circuiti

Definendo con $\mathcal{C}$ la collezione di tutti i circuiti ammissibili per un veicolo e con γ_C il costo di un circuito $C \in \mathcal{C}$ (vale a dire la somma dei costi degli archi nel circuito), indicando con $\mathcal{C}_i$ la sottocollezione dei circuiti in $\mathcal{C}$ che visitano il cliente i, ed introducendo le variabili:

$$x_C := \begin{cases} 1, & \text{se la soluzione contiene il circuito C} \\ 0, & \text{altrimenti} \end{cases}$$

il corrispondente ILP ha la forma:

$$\min \sum_{C \in \mathcal{C}} \gamma_C x_C \tag{143}$$

$$\sum_{C \in \mathcal{C}} x_C = k, \tag{144}$$

$$\sum_{C \in \mathcal{C}_i} x_C = 1, \quad i \in V \setminus \{0\} \tag{145}$$

$$x_C \in \{0,1\}, \quad C \in \mathcal{C} \tag{146}$$

Si osservi che, valendo la disuguaglianza triangolare, i vincoli (145) potrebbero equivalentemente essere scritti nella forma "$\geq$"

Inoltre, dato che il costo γ_C di un circuito $C \in \mathcal{C}$ non è costante ma dipende dal circuito stesso, a differenza dei problemi del Bin Packing e del Vertex Coloring *non è corretto* in questo caso limitarsi a considerare circuiti che visitino un sottoinsieme di clienti massimale (rispetto al vincolo di capacità)

La considerazione sopra vale in generale per i modelli di questo tipo: se i costi delle variabili non sono tutti uguali non è corretto limitarsi a considerare variabili associate a sottoinsiemi massimali nella collezione

Il problema di generazione di colonne per il rilassamento continuo del modello (ottenuto sostituendo il vincolo che le x siano binarie con quello che siano non-negative) richiede la definizione del suo problema duale

Si indichi con I_C l'insieme dei clienti visitati da un circuito $C \in \mathcal{C}$ e si introducano le variabili z associate all'equazione (144) ed y_i associate alle equazioni (145) (tutte le variabili duali possono essere negative)

Il duale ha la forma:

$$\max kz + \sum_{i \in V \setminus \{0\}} y_i \tag{147}$$

$$z + \sum_{i \in I_C} y_i \leq \gamma_C, \quad C \in \mathcal{C} \tag{148}$$

Il problema di generazione di colonne richiede, dati z^* e $y^* = (y_i^*)$, di determinare, se esiste, un circuito $C \in \mathcal{C}$ tale che:

$$z^* + \sum_{i \in I_C} y_i^* > \gamma_C \Leftrightarrow \sum_{i \in I_C} y_i^* - \sum_{a \in C} c_a > -z^*$$

Un tale circuito C esiste se e solo se il valore ottimo dell'ILP che segue è $> -z^*$ (ed in tal caso il circuito è indicato dalla soluzione dell'ILP)

Si considerino le variabili:

$$w_i := \begin{cases} 1, & \text{se il vertice } i \text{ è visitato dal circuito } C \\ 0, & \text{altrimenti} \end{cases}$$

e

$$u_a := \begin{cases} 1, & \text{se l'arco } a \text{ appartiene al circuito } C \\ 0, & \text{altrimenti} \end{cases}$$

Il modello ILP per la generazione di colonne, analogo al modello 2 ma in questo caso riferito ad un solo circuito, è:

$$\max \sum_{i \in V \setminus \{0\}} y_i^* w_i - \sum_{a \in A} c_a u_a \tag{149}$$

$$\sum_{a \in \delta^-(i)} u_a = w_i, \quad i \in V \setminus \{0\} \tag{150}$$

$$\sum_{a \in \delta^+(i)} u_a = w_i, \quad i \in V \setminus \{0\} \tag{151}$$

$$\sum_{a \in \delta^-(0)} u_a = 1, \tag{152}$$

$$\sum_{a \in \delta^+(0)} u_a = 1, \tag{153}$$

$$\sum_{i \in V \setminus \{0\}} d_i w_i \leq b \tag{154}$$

$$\sum_{a \in \delta^+(S)} u_a \geq w_i, \quad S \subseteq V \setminus \{0\},\ i \in S \tag{155}$$

$$u_a, w_i \in \{0,1\}, \quad a \in A,\ i \in V \setminus \{0\} \tag{156}$$

Taglio e Impaccamento Multidimensionale

Anche se hanno un campo di applicazioni profondamente diverso, i problemi di taglio ed i problemi di impaccamento hanno gli stessi modelli matematici

I problemi di taglio/impaccamento multidimensionale richiedono di ritagliare/impaccare un dato insieme di oggetti multidimensionali da/in un dato insieme di basi/contenitori multidimensionali

Esempi sono dati da:

- Taglio di indumenti da un rotolo di stoffa
- Taglio di finestre da lastre di vetro
- Taglio di pannelli da lamine
- Impaccamento di pallet in contenitori

Nella maggior parte dei casi, i problemi di taglio sono bidimensionali ed i problemi di impaccamento tridimensionali, anche se per questi ultimi il numero di dimensioni si riduce se, lungo una dimensione, tutti gli oggetti sono uguali (ad esempio parallelepipedi con altezze tutte identiche)

Per semplicità e concretezza, ci limiteremo a considerare problemi di impaccamento bidimensionali

Generalmente gli obiettivi si suddividono in:

- Minimizzare lo "spazio" utilizzato per impaccare tutti gli oggetti
- Massimizzare il profitto degli oggetti impaccati in uno "spazio" finito

anche se sono ovviamente possibili obiettivi ottenuti combinando i due sopra

I problemi *base* in questo ambito, a cui si fa riferimento anche per risolvere problemi più complessi, corrispondono al caso in cui:

1. Ogni oggetto i è un *rettangolo* di *base* w_i e *altezza* h_i, $i = 1, \ldots, m$
2. Ogni contenitore è un *rettangolo* di *base* e *altezza* dati
3. Ogni oggetto impaccato in un contenitore deve avere la base parallela alla base del contenitore (impaccamento *ortogonale senza rotazione*)

I modelli che vedremo si adattano facilmente anche ai casi in cui gli oggetti ed i contenitori hanno forme qualsiasi (anche se la loro rappresentazione diviene in questo caso notevolmente più complessa) mentre devono essere modificati sostanzialmente nel caso in cui il vincolo 3 non sia imposto (soprattutto nel caso in cui sia consentita una rotazione di in angolo qualsiasi anzichè solo di 90°)

I problemi base sono in questo caso:

- *Knapsack Bidimensionale*, in cui è dato un unico contenitore quadrato di base ed altezza L, ogni oggetto i ha un profitto p_i, e si vuole impaccare un sottoinsieme di oggetti di profitto massimo nel contenitore

- *Bin Packing Bidimensionale*, in cui sono dati infiniti contenitori quadrati di base ed altezza L e si vogliono impaccare tutti gli oggetti nel minimo numero di contenitori
- *Strip Packing Bidimensionale*, in cui è dato un contenitore rettangolare di base L ed altezza infinita e si vogliono impaccare tutti gli oggetti minimizzando l'altezza utilizzata (corrispondente alla massima altezza toccata da un oggetto nel contenitore)

Si osservi che Kanpsack e Bin Packing Bidimensionali sono generalizzazioni dei corrispondenti problemi in una dimensione, mentre il corrispettivo dello Strip Packing Bidimensionale in una dimensione è banale (impaccare segmenti di data lunghezza in una semiretta)

Tutti e tre i problemi sono NP-completi (in particolare, è NP-completo il problema di decidere se un dato insieme di retangoli può essere impaccato in un quadrato) e difficili in pratica

Un Modello ILP per il Knapsack Bidimensionale

In questo ambito vedremo un modello matematico per il Knapsack Bidimensionale che si adatta immediatamente al caso degli altri due problemi (aggiungendo un indice di bin alle variabili nel caso del Bin Packing Bidimensionale)

Nel modello illustrato le variabili non si limitano a specificare se un oggetto viene impaccato o meno (come è sufficiente fare nel caso monodimensionale), ma indicano anche *in quale posizione* un oggetto viene impaccato

Si assumerà che le dimensioni degli oggetti e del contenitore siano tutti valori interi, in modo da potere assumere senza perdita di generalità che le coordinate dei vertici di tutti gli oggetti impaccati siano anch'esse intere

Indicando le coordinate avendo come origine $(0, 0)$ l'angolo *in basso a sinistra* del contenitore, le variabili utilizzate dal modello sono:

$$x_{ijk} := \begin{cases} 1, & \text{se l'oggetto } i \text{ è impaccato con l'angolo in basso a sinistra nelle coordinate } (j, k) \\ 0, & \text{altrimenti} \end{cases}$$

Per ogni oggetto i, per garantire che le variabili utilizzate corrispondano ad un impaccamento che non esce dal contenitore, vengono introdotte le variabili x_{ijk} per $j = 0, \ldots, L - w_i$ e $k = 0, \ldots, L - h_i$

Inoltre, per maggiore leggibilità, vengono introdotte anche le seguenti variabili, non necessarie in quanto uguali ad una somma di variabili x:

$$y_i := \begin{cases} 1, & \text{se l'oggetto } i \text{ è impaccato} \\ 0, & \text{altrimenti} \end{cases}$$

Il modello ha la forma:

$$\max \sum_{i=1}^{m} p_i y_i \tag{157}$$

$$\sum_{j=0}^{L-w_i} \sum_{k=0}^{L-h_i} x_{ijk} = y_i, \quad i = 1, \ldots, m \tag{158}$$

$$x_{ijk} + x_{lqr} \leq 1, \quad x_{ijk}, x_{lqr} \text{ "incompatibili"} \tag{159}$$

$$y_i, x_{ijk} \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 0, \ldots, L - w_i,\ k = 0, \ldots, L - h_i \tag{160}$$

dove due variabili x_{ijk} e x_{lqr} sono "incompatibili" se corrispondono al posizionamento dei due oggetti i e l *sovrapposti* nel contenitore, formalmente:

$$((j \leq q < j + w_i) \textbf{ or } (q \leq j < q + w_l)) \textbf{ and } ((k \leq r < k + h_i) \textbf{ or } (r \leq k < r + h_l))$$

Eliminando le variabili y (cioè sostituendole con la somma di variabili x corrispondente), il modello indicato è chiaramente il modello "debole" (38)-(40) per il problema di Stable Set associato al grafo in cui i vertici corrispondono alle variabili x (ciascuno con profitto pari al profitto dell'oggetto associato alla variabile) ed i lati alle incompatibilità tra coppie di variabili x

In altre parole, utilizzando variabili che specificano le coordinate degli oggetti impaccati, il Knapsack Bidimensionale è formulato come caso particolare dello Stable Set

Per ottenere un modello più forte, si può ovviamente analizzare la struttura del grafo e derivare vincoli associati a clique massimali da utilizzare al posto di (159)

In alternativa, non considerando la struttura del grafo ma il problema geometrico che si sta rappresentando, si possono derivare i seguenti vincoli, da utilizzare al posto di (159)

Si consideri il quadrato di lato unitario e con angolo in basso a sinistra di coordinate (j, k): nella soluzione al massimo uno degli oggetti impaccati potrà sovrapporsi a tale quadrato

Per un oggetto i, le variabili x_{iqr} corrispondenti alla sovrapposizione dell'oggetto al quadrato di lato unitario e con angolo in basso a sinistra di coordinate (j, k) sono quelle corrispondenti ai valori di q nell'intervallo:

$$\{\max\{0, j - w_i + 1\}, \ldots, \min\{L - w_i, j\}\}$$

e di r nell'intervallo:

$$\{\max\{0, k - h_i + 1\}, \ldots, \min\{L - h_i, k\}\}$$

Corrispondentemente, i vincoli che esprimono la condizione che, per ogni quadrato di lato unitario nel contenitore, al massimo un oggetto impaccato si sovrapponga a tale quadrato hanno la forma:

$$\sum_{i=1}^{m} \sum_{q=\max\{0,j-w_i+1\}}^{\min\{L-w_i,j\}} \sum_{r=\max\{0,k-h_i+1\}}^{\min\{L-h_i,k\}} x_{iqr} \leq 1, \quad j,k = 0,\ldots,L-1 \tag{161}$$

e vengono usati al posto di (159)

Esempio 17 Si consideri il caso $m = 5$, $L = 10$, $w = h = (7,5,4,4,2)$ (ovverosia gli oggetti sono tutti quadrati) e si scriva il vincolo (161) associato a $j = 3$ e $k = 4$

Il Caso più Frequente in Pratica

Nella maggior parte dei casi pratici il problema da risolvere è un problema di Bin Packing Bidimensionale

Anzichè formularlo tramite la variante menzionata del modello sopra, il problema viene formulato tramite un modello del tipo (37), in cui $\mathcal{S}$ rappresenta la collezione dei sottoinsiemi massimali di oggetti (rettangoli) che possono essere impaccati in un bin (quadrato), ed il modello illustrato per il Knapsack Bidimensionale risolve il corrispondente problema di generazione di colonne (con profitti degli oggetti dati dalle variabili duali associate)

Scheduling

I problemi di *scheduling* si riferiscono all'organizzazione temporale del lavoro che deve essere svolto in un generico processo produttivo

Data l'ampia gamma di situazioni reali possibili, il numero di problemi di scheduling di interesse pratico è molto vasto, e non è possibile individuare un problema "base" di riferimento

Per consentire l'identificazione dei problemi all'interno di una classe così vasta, è stata introdotta addirittura una sintassi apposita che specifica le caratteristiche fondamentali di ciascun problema (tale sintassi non verrà illustrata qui)

In linea di massima, in un problema di scheduling sono dati n *lavori* che devono essere eseguiti da m *macchine*, e si deve decidere l'assegnazione dei lavori alle macchine e/o l'ordine in cui ciascuna macchina esegue i lavori assegnatile

L'istante a partire da cui le macchine possono lavorare è convenzionalmente fissato a 0, e ciascuna macchina può eseguire un solo lavoro alla volta

Ciascun lavoro j è caratterizzato da:

- un insieme di *operazioni* in cui è suddiviso (eventualmente *una sola*, nel qual caso il lavoro e la sua operazione sono la stessa cosa)

- il tempo di esecuzione (*processing time*) di ciascuna operazione su ciascuna macchina (eventualmente *infinito* se la macchina non può eseguire l'operazione)
- un istante r_j prima del quale non può essere iniziato (*release date*) (eventualmente 0)
- un istante d_j entro il quale dovrebbe essere terminato (*due date*) (eventualmente ∞)

Le macchine possono essere:

- *identiche*, ovverosia qualunque operazione può essere eseguita da ciascuna macchina nello stesso tempo
- *uniformi*, ovverosia ciascuna ha una sua *velocità* e qualunque operazione può essere eseguita da ciascuna macchina in un tempo inversamente proporzionale alla velocità della macchina
- *differenti*, ovverosia il tempo di esecuzione di una operazione cambia da macchina a macchina (e spesso l'operazione può essere eseguita solo da alcune macchine)

Caratteristiche aggiuntive dei problemi possono essere:

- la possibilità di *interrompere* l'esecuzione di un'operazione riprendendola successivamente (eventualmente su una machina diversa), detta *preemption*
- l'esistenza di *vincoli di precedenza* tra operazioni di uno stesso lavoro e di lavori diversi

Gli obiettivi sono generalmente dati dalla *minimizzazione* della somma di o del massimo tra:

- *istante di completamento* C_j del lavoro j
- *lateness* $L_j := C_j - d_j$ del lavoro j (penalizzando il ritardo rispetto alla due date e premiando l'eventuale anticipo)
- *tardiness* $L_j := \max\{0, C_j - d_j\}$ del lavoro j (penalizzando il ritardo rispetto alla due date ma non premiando l'eventuale anticipo)

L'obiettivo di gran lunga più frequente è la minimizzazione del massimo istante di completamento di un lavoro, detto *makespan* $C_{\max} := \max_{j=1}^{n} C_j$ – in altre parole si vogliono terminare tutti i lavori il prima possibile

Nel seguito verranno illustrati due esempi rilevanti, il secondo dei quali fonirà lo spunto per discutere il fatto che, per molti problemi di scheduling di interesse pratico, non si conoscono modelli ILP "forti"

L'effetto del non conoscere modelli ILP "forti" ha come conseguenza che, per molti problemi di interesse pratico, non solo non si conosce il valore della soluzione ottima, ma non si conoscono neppure lower bound "buoni" su di essa che possano certificare la qualità delle soluzioni euristiche determinate

Macchine Identiche Parallele, o $P||C_{\max}$

Il problema delle *Macchine Identiche Parallele*, indicato con $P||C_{\max}$ secondo la sintassi menzionata sopra, corrisponde al caso in cui ciascun lavoro j ha una sola operazione p_j con tempo di esecuzione p_j su una qualunque macchina (oltre ad avere $r_j = 0$ e $d_j = \infty$), e si vogliono assegnare i lavori alle macchine in modo da minimizzare il makespan

Esempio 18 Si consideri il caso $m = 2$, $n = 4$, $p = (3, 1, 4, 3)$ e si determini la soluzione ottima

Per definire un modello ILP per il problema, si osservi innanzitutto che l'unica decisione da prendere corrisponde all'assegnazione dei lavori alle macchine e *non* all'ordine in cui le macchine eseguono i lavori assegnati loro, in quanto il makespan non dipende da quest'ultimo, essendo dato dalla massima somma dei tempi di esecuzione dei lavori assegnati ad una macchina

Un'interpretazione possibile del problema visto è la variante del Bin Packing in cui il numero m di bin utilizzati è fissato mentre si può variare la capcità b, per la quale si vuole trovare il valore *minimo* che garantisce che gli oggetti siano impaccabili negli m bin

Introducendo le variabili:

$$x_{ij} := \begin{cases} 1, & \text{se il lavoro } j \text{ è assegnato alla macchina } i \\ 0, & \text{altrimenti} \end{cases}$$

un modello con funzione obiettivo nonlineare è dato da:

$$\min \max_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{162}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n \tag{163}$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m, \; j = 1, \ldots, n \tag{164}$$

Come per molti altri problemi di scheduling (ed anche altri problemi di tipo "min max") un modello ILP è ottenuto introducendo la variabile z e sostituendo (162) con:

$$\min z \tag{165}$$

$$\sum_{j=1}^{n} p_j x_{ij} \quad \leq \quad z, \quad i = 1, \ldots, m \tag{166}$$

Il modello ILP visto ha tutti i difetti del modello ILP "debole" per il Bin Packing

A differenza del Bin Packing non esiste un modello "forte" in questo caso, e la cosa migliore da fare per risolvere il problema è determinare il valore ottimo di z provando diversi valori e verificando che siano ammissibili risolvendo il corrispondente modello di bin packing con capacità z tramite il modello "forte" (la sequenza di valori di z è guidata dalla cosiddetta *ricerca binaria*, non illustrata qui)

Flow Shop, o $F||C_{\max}$

Il problema del *Flow Shop*, indicato con $F||C_{\max}$ secondo la sintassi menzionata sopra, appartiene alla famiglia dei problemi di tipo "Shop" (che include il *Job Shop* e l'*Open Shop*) che, tra i problemi combinatori facili da formulare, sono tra i più difficili da risolvere

Nel Flow Shop, ciascun lavoro j è suddiviso in m operazioni, l'i-esima delle quali deve essere eseguita sulla macchina i, ha tempo di esecuzione p_{ij}, e può iniziare solo dopo che sia stata completata l'$i-1$-esima (la prima operazione puó iniziare all'istante 0, cioè $r_j = 0$; inoltre $d_j = \infty$)

Si vuole decidere, per ogni macchina, l'ordine di esecuzione delle operazioni dei vari lavori in modo da minimizzare il makespan

Esempio 19 Si consideri il caso $m = n = 3$, $p = \begin{pmatrix} 5 & 4 & 2 \\ 8 & 7 & 4 \\ 3 & 6 & 5 \end{pmatrix}$ e si determini una soluzione euristica

Introducendo le variabili continue:

t_{ij} := istante di inizio dell'i-esima operazione del lavoro j (sull'i-esima macchina)

il modello più naturale per il problema, con vincoli nonlineari, è dato da:

$$\min z \tag{167}$$
$$t_{mj} + p_{mj} \leq z, \quad j = 1, \ldots, n \tag{168}$$
$$t_{(i-1)j} + p_{(i-1)j} \leq t_{ij}, \quad j = 1, \ldots, n,\ i = 2, \ldots, m \tag{169}$$
$$t_{ij} + p_{ij} \leq t_{ik} \quad \textbf{or} \quad t_{ik} + p_{ik} \leq t_{ij}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{170}$$
$$t_{1j} \geq 0, \quad j = 1, \ldots, n \tag{171}$$

dove i vincoli (169) impongono l'ordine di esecuzione delle operazioni di un lavoro ed i vincoli nonlineari (170) impongono che ciascuna macchina i esegua al più una operazione alla volta (l'i-esima operazione del lavoro j viene eseguita prima dell'i-esima operazione del lavoro k, *oppure* l'i-esima operazione del lavoro k viene eseguita prima dell'i-esima operazione del lavoro j)

Per linearizzare i vincoli (170) occorre, come nel caso delle finestre temporali per il problema del Vehicle Routing, introdurre variabili binarie che esprimano la precedenza tra le operazioni di una stessa macchina ed un coefficiente "big M":

$$x_{ijk} := \begin{cases} 1, & \text{se l'}i\text{-esima operazione del lavoro } j \text{ precede l'}i\text{-esima operazione del lavoro } k \\ 0, & \text{altrimenti} \end{cases}$$

sostituendo (170) con:

$$x_{ijk} + x_{ikj} = 1, \quad j, k = 1, \ldots, n,\ j < k,\ i = 1, \ldots, m \tag{172}$$
$$t_{ij} + p_{ij} \leq t_{ik} + M x_{ikj}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{173}$$
$$x_{ijk} \in \{0, 1\}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{174}$$

La presenza del "big M" rende estremamente "debole" il rilassamento continuo del modello visto, ma d'altra parte per il problema del Flow Shop (così come per gli altri problemi di tipo Shop e per molti altri problemi di Scheduling) non si conoscono modelli ILP "forti"